1/5

PERSONAL COMPUTER WORLD

50p U.S.$2.00

SEPTEMBER 1978

Europe's first magazine for personal computers for home and business use

- GETTING IT TOGETHER
- THE SOFT FAÇADE
- 'A TIDAL WAVE'
- BUS OF THE CENTURY
- BUZZWORDS

Precisely speaking... programme your way to a prize!

So your'e interested in personal computing?

Our own PCW show...news inside

TANDY TRS80
The Great Leveller

PERSONAL COMPUTER WORLD

Europe's first magazine for personal computers for home and business use

UK 50p
US $2.00

Vol 1, No 5
September 1978

CONTENTS

Editorial and Advertising Office:
62A Westbourne Grove, London W2
Phone: 01-229 5599 (Publisher)
01-727 8758 (Editorial and Advertising)

Publisher:
A. Zgorelec
Editor:
Meyer N. Solomon
Editorial Assistant:
Roger C. Wilkins
Policy Advisor:
Peter Crofton-Sleigh, FRAS

Consultants:
John Coll, Mike Dennis, Neil Harrison, Charles Sweeten, Patrick Sutton, Michael James, R.W. Davy, David Hebditch, Sheridan Williams.
Art: **Sauveur Laurent Sant, Kathryn Hamme**
Secretarial:
Vanessa Blackburn Kiddle
Layout Consultants:
D. Norris, T. Gabos
PCW Photography: **Yoshi Imamura, Peter McGee**
Typesetting & Artwork: **Gilfillan**
Cover & Photograph **Y. Imamura taken at Tandy store, Bayswater, Courtesy of D. Malekout (Manager), and Ed. Morris (Asst. Manager).**

CONTRIBUTORS:

We welcome interesting articles written simply and clearly. You need not be a specialist to write for us. MS should not be more than 3000 words long, lines double spaced, with wide margins. Line drawings and photographs wherever possible. Enclose a stamped self-addressed envelope if you would like your article returned.

Manufacturers, suppliers and dealers are welcome to contribute technical articles, and send product information, but we are pledged to an independent viewpoint and will publish evaluations and reasoned criticism or praise, space permitting. Naturally there will be right of reply. Views expressed in articles are not necessarily those of Personal Computer World.

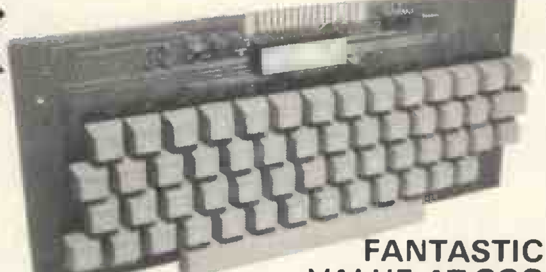
We may make arrangements to offer our readers products at special prices, for a limited period, in line with the policy outlined above.

Published monthly by Intra Press, 62A Westbourne Grove, London W2. Phone: 01-229 5599. Contents fully protected by copyright. All rights reserved. Subscription rates: Britain £8 for 12 issues. Prices include postage. USA – $10 for six issues, $20 for 12 issues. Continent and elsewhere: £9.80 for twelve issues. Prices include postage.
Printed by Carlisle Web Offset, 55 Conduit Street, Newtown Trading Estate, Carlisle CA2 7NR. Sole UK Distributors: Seymour Press Ltd., 334 Brixton Road, London S.W.9., England. Distribution to specialist shops by Intra Press.

NEW

56-STATION ASCII KEYBOARD

NOW IN STOCK — ASCII KEYBOARD MODEL KB756



Full 128 character set with ROM encoder (Upper and lower case + control shift).
Fully TTL — compatible — power requirements + 5v-12v.
Supplied with full technical data, code chart, pin connections, circuit diagram and application notes.

FANTASTIC VALUE AT £60

+ £1.50 p&p + 8% VAT (mail order total £66.42)

SELF-SCAN ALPHANUMERIC PANEL DISPLAY	BALL MIRATEL VIDEO MONITOR
16/18 position display with 64 character repertoire, 5 x 7 dot matrix. Input 6-bit BCD-code, power requirements + 5v, -12v. Character size 0.40" x 0.28". Overall dimensions 8¼" x 2¼" x 1¼". Supplied with full technical data. Price £55.00 + 75p P&P + 8% VAT (Mail order total £60.21)	9" diagonal P4 phosphor tube. Bandwidth 12 MHz (-3dB). Input voltage 220V 50/60 Hz 24W. Out put voltage + 15V DC (short circuit protected) + 12kV DC; 12.6V rms. Supplied complete with high & low voltage power supplies, amplifier, and attractive moulded plastic housing including space for keyboard. Price £95.00 + carriage + VAT.

SPECIAL PURCHASE - MITE

123P Alphanumeric printer mechanisms

Solenoid-operated page printer using standard reversible typewriter ribbon. Prints standard 64-ASCII character set on 8½" paper (80 characters per line, 6 lines to the inch). Maximum speed 11 cps. Power requirements 115VDC. Compact, light-weight unit 9¾ lbs, 12" x 9" x 2¾". Supplied complete with full technical manual.

BRAND NEW SURPLUS
ONLY £75
+ £3.50 P&P + 8% VAT
(Mail order total £84.78)

We also specialise in: DEC minis — PDP8 and PDP11 processors, add-on memory, peripherals and spares. Hard copy terminals — ASR 33 and KSR 33 Teletypes, Data Dynamics 390, Texas Silent 700. Send for complete lists.

ELECTRONIC BROKERS LTD.
49-53 Pancras Road, London NW1 2QB.
Tel: 01-837 7781. Telex: 298694.

HAZELTINE VISUAL DISPLAY UNIT

* Teletype Compatible
* 12" Diagonal Screen
* TTY Format Keyboard
* 12 lines of 80 characters
* 64 ASCII Character Set
* 5 x 7 Dot Matrix
* Switch-selectable Transmission Speeds up to 9600 baud
* Switch-selectable Parity
* Standard CCITT V.24 Interface

MODEL H-1000 PRICE £350 + carriage and VAT

Also available: —
Model H-1200: Specification as for H-1000 except 24 lines of 80 characters displayed. Price £425.00 + carriage + VAT.
Model H-2000: Buffered/Editing model with direct cursor addressing, dual intensity video, and detachable keyboard with separate numeric and edit clusters. 27 lines of 74 characters. Price £495.00 + carriage + VAT.

A copy of trading conditions supplied on request

ANNOUNCING THE MICROS

JOIN THE MICRO REVOLUTION

Access — We welcome Access · master charge · EUROCARD

£399 for a Z80 based microcomputer, built and tested

Designed for educational establishments, personal computing and small business users

* Includes 1K monitor Eprom, 47 key solid state keyboard, video, TV, cassette and teletypewriter interfaces, serial i/o, 2 parallel i/o ports, 2K bytes RAM, power supplies and instrument housing.
* Connect to domestic TV or video monitor to complete the system
* 48 x 16 character video matrix
* 47 key contactless ASC11 keyboard
* Hard copy on teletypewriter
* 2 TTL compatible parallel i/o ports
* RS232 serial i/o port
* Load and dump programmes on unmodified cassette recorder
* Up to 16K byte mixed RAM and Eprom in table top housing
* Expandable up to 64K bytes
* Security locked power switch
* British designed and built
* Available in kit form for £360
* Credit terms available

THE MICRONICS COMPANY
1, STATION ROAD TWICKENHAM MIDDLESEX
8972 7044

PART OF THE MICRO REVOLTION
Prices exclusive of VAT and carriage

Look To The North Star HORIZON Computer.

HORIZON – a complete, high performance microprocessor system with integrated floppy disk memory. HORIZON is attractive, professionally engineered, and ideal for business, educational and personal applications.

To begin programming in extended BASIC, merely add a CRT, teletype or other hard-copy terminal. HORIZON-1 includes a Z80A processor, 16K RAM, minifloppy disk and 12-slot S-100 motherboard with serial terminal interface – all standard equipment.

WHAT ABOUT PERFORMANCE?

The Z80A processor operates at 4MHz – double the power of the 8080. The RAM memory board lets the Z80A execute at full speed. HORIZON can load or save a 10K byte disk program in less than 2 seconds. Each diskette can store 90K bytes.

AND SOFTWARE, TOO

HORIZON includes the North Star Disk Operating System and full extended BASIC on diskette ready at power-on. This BASIC, now in widespread use, has virtually everything desired in a BASIC, including sequential and random disk files, formatted output, a powerful line editor, strings, machine language CALL and more. Optional software (under CP/M) includes – CBASIC compiler/interpreter BASIC, Microsoft Disk Extended BASIC, MAC Macro Assembler, Microsoft COBOL-80 and FORTRAN-80, and more.

EXPAND YOUR HORIZON

Also available – Hardware floating point board (FPB); additional 8K and 16K memory boards. Add a second disk drive and you have HORIZON-2. Economical serial and parallel I/O ports may be installed on the motherboard. Many widely available S-100 bus peripheral boards can be added to HORIZON.

QUALITY AT THE RIGHT PRICE

HORIZON Z80A processor board, RAM, FPB and MICRO DISK SYSTEM can be bought separately for either Z80 or 8080 S-100 bus systems.

HORIZON-1: £1,275 assembled and tested.
HORIZON-2: £1,550 assembled and tested.

8K, 250ns static RAM: £145; 16K, 250ns static RAM: £295; FPB: £215; Z80A board: £185; 2nd serial I/O: £45; parallel I/O: £45. All prices are for assembled and tested units. Prices are exclusive of V.A.T. and carriage, and are subject to change.

For full details contact:

INTERAM

INTERAM Computer Systems Ltd.
59 Moreton Street
Victoria, London SW1V 2NY
Telephone: 01-834 0261/2733

TRS-80—THE BIGGEST NAME IN LITTLE COMPUTERS

Complete and Ready to Go NOW!

- Amazing all-in system at a low price
- Big 12" video monitor
- Standard typewriter Keyboard

The TRS-80 computer system is 100% wired, and tested for 240 VAC so you can put it to work immediately! It's ideal for finances, education, accounting, laboratory—even games at home. Below are four complete TRS-80 systems incorporating different combinations of RAM (4K and 16K) and ROM (Level-1 and Level-11 BASIC). Choose the one that's right for you. Expansion is easy due to TRS-80's modular design. All TRS-80 systems below include a 12" video monitor, Realistic CTR-41 battery/AC cassette recorder, power supply, 232-page user's manual, and a 2-game cassette.

Level-1 BASIC system

4K ROM, 4K RAM
Cat.No. 26-1001, 26-1201, 14-841 **£499**

Level-11 BASIC system

12K ROM, 4K RAM
Cat.No. 26-1004, 26-1201, 14-841 **£578**

Level-1 with 16K RAM

4K ROM, 16K RAM
Cat.No. 26-1003, 26-1201, 14-841 **£728**

Level-11 with 16K RAM

12K ROM, 16K RAM
Cat.No. 26-1006, 26-1201, 14-841 **£807**

You can see the TRS-80 at these Tandy Stores and Dealerships

BASINGSTOKE	22 London Street, Basingstoke. Tel: 52795
BIRMINGHAM AREA	Bilston Road, Wednesbury. Tel: (021) 556 6429
	528 The Bridge, Bull Ring Shopping Centre, Birmingham. Tel: (021) 643 3876
	57-58 Dale End, Birmingham. Tel: (021) 236 4744
BOLTON	5 Nelson Square, Bolton. Tel: 386538
BOURNEMOUTH	134 Commercial Road, Bournemouth. Tel: 293606
BRIGHTON	70 London Road, Brighton. Tel: 693446
BRISTOL	5 Badminton Road, Downend, Bristol. Tel: 561917
BROMSGROVE	6-8 St. John Street, Bromsgrove. Tel: 71800
CAMBRIDGE	12 High Street, Cambridge. Tel: 68155
CHESTER	Kwik Save Centre, Sealand Road, Chester. Tel: 375794
COVENTRY	4 Hales Street, Coventry. Tel: 22894
DARLINGTON	15-16 Priestgate, Darlington. Tel: 58676
DERBY	33 Victoria Street, Derby. Tel: 371066
DONCASTER	32-34 Kingsgate, Waterdale Centre, Doncaster. Tel: 21992
GLASGOW	Audio Visual, 340 Argyle Street, Glasgow. Tel: (041) 221 8958
GLOUCESTER	47 Kings Square, Clarence Street, Gloucester. Tel: 31323
LEEDS	72 Merrion Centre, Leeds. Tel: 42520
LIVERPOOL	168 St. John's Centre, Market Way, L'pool. Tel: (051) 708 0161
LONDON AREA	The Colonnades, Porchester Road, Queensway, Bayswater, W.2. Tel: (01) 221 5317
	7 Embassy Court, Welling. Tel: (01) 303 5483
	124-126 The Broadway, Wimbledon, S.W.19. Tel: (01) 542 6389
	6 New Broadway, S.W.5. Tel: (01) 579 1320
	21 Sentinel Square, Hendon, N.W.4. Tel: (01) 202 7331
LUTON	157 Dunstable Road, Luton. Tel: 36159
MANCHESTER	4-8 The Mall Shopping Centre, Hyde. Tel: (061) 368 0268
	13-15 The Mall Shopping Centre, Sale. Tel: (061) 973 0371
	The Arndale Centre, Stretford. Tel: (061) 865 8214
NEWCASTLE-UPON-TYNE	23 Newgate Centre, Newgate, Newcastle. Tel: 21478
NORTHAMPTON	Weston Favel Shopping Centre, Northampton. Tel: 326354
NOTTINGHAM	126-128 Front Street, Arnold, Nottingham. Tel: 202626
SUTTON	206 High Street, Sutton, Surrey. Tel: (01) 643 8687
SWANSEA	Radio Supplies, 80 Gower Road, Sketty, Swansea. Tel: 24140
WITNEY	Witney Audio, 29 Corn Street, Witney. Tel: 2414
WOLVERHAMPTON	1 Market Street, Wolverhampton. Tel: 21148

Or contact Computer Sales Department, Tandy Corporation, Bilston Road, Wednesbury, West Midlands. WS10 7JN. Tel: (021) 556 6101

TANDY

WORLDWIDE ELECTRONICS

FROM THE PUBLISHER AND EDITOR

As space is at a premium in this issue, we have decided to just give our readers this message:

The Personal Computer World Show will be the first of its kind in Europe. People will turn up to enjoy themselves, look, listen, learn; be fascinated by computers playing chess against each other under the supervision of the International Chess Master, David Levy; look at the displays in the Innovators' Corner; see a programmed model railway by CAP Microsoft; and meet Leslie Solomon (Technical Director of *Popular Electronics* and-really-no-relative of the editor!). Leslie is bringing over a couple of marvellous devices.

Each day has been carefully planned. Our exhibitors know that the majority of our readers are people who one way or another are shaping or will shape the future of computing (yes, computing and not specially personal computing) in Europe. Our conference speakers know that delegates will come with open minds and with two motives: to learn and to interact.

We would like here to say how privileged we feel that Julia Howlett, of the National Physical Laboratories, will be one of our speakers. Julia Howlett, who is blind, is the project leader of MAVIS, a microprocessor based aid for the disabled.

PCW is looking forward to our readers attending their own Show in their thousands.

Fuller details of the Show are given elsewhere in this issue.

PCW Book Review

REVIEW OF "TALES FROM THE COMPUTER ROOM"

by John Race.
(Published by the Author and printed by Brunel University.
On sale at University Bookshops or by post (65p + 10p pp)
from The Bookshop, Brunel University, Uxbridge, Middlesex.

Computer programming is arguably one of the most alienated activities man does in the course of normal work. Those well steeped in computing seem to take an uncommon delight in talking shop, swapping jargon and indulging in pointless gee-whizzery. It therefore comes as a pleasant surprise to see the fevered brain of the computer person turning to creative writing in the "new" field of computer fiction.

John Race unfortunately uses too many buzz words for his writing to be accessible to those unfamiliar with computers. I don't know COBOL so the tale "On Mars, sloppy COBOL is Sudden Death", despite its vivid imagery, fell rather flat with me, even though an explanation of the theme is included as a post script.

Other stories are very entertaining sci-fi. They all have an element of humour and the absurd, inspired presumably by the real absurdities of computer systems. The science is dubious, but intentionally so I think, to make for humour. All are highly imaginative, ranging from murder with the powered sliding door of a tape deck, to a personality trapped inside a system with no access to peripherals, reduced to pleading in each core dump for someone to press reset.

Then, two tales of programmers who have written their employers' software so as to make themselves absolutely indispensible are the closest John Race gets to making social comment.

I hope that this thin volume spawns more computer fiction, but I would like to see it more down to earth, serving to communicate rather than being more sophisticated chatter between technocrafts.

A good buy for those fairly conversant with computing.

Roger Wilkins

MODERN GUIDE TO DIGITAL LOGIC' PROCESSORS' MEMORIES & INTERFACES, 1976

289 pages. (Foulsham Tab, paper, 5½" x 8½" £3.60)

The subtitle of this book is misleading. The reader might expect to find a book crammed with information on (micro) processors, memories and interfaces. In fact out of ten chapters only one deals with microprocessors, one with memories and if 'interfaces' is interpreted in the sense of computer/peripheral interface, none on interfaces!

The eight pages (chapter 9) on microprocessors are very vague and introductory. Perhaps this is a reflection of the date of publication. The chapter on memories is better, but is very technical, with such things as cost/performance comparisons of memory techologies being examined at length. The sort of information of use to the amateur (eg. how big, how fast, and how much) is not included. Interfacing is covered in the sense of interfacing different logic families, e.g. TTL to CMOS, and as such is very useful, but not what one might have expected from the subtitle.

The rest of the book consists of fairly academic chapters on various logic families, their advantages and disadvantages, e.g. MECL 10,000 v Schottky TTL, Designing with MECL 10,000, etc. As the only two logic families that the amateur comes into contact with are CMOS and TTL (sometimes Schottky TTL) these arguments are fairly irrelevant.

This book is for the professional designer or the university academic, not the home computer or electronics enthusiast, no matter what impression might be gained from the title or the introduction.

Michael James

Letters

An Alternative Programmer

Your article 'Putting Bits in Their Place' by David Goadby which appeared in the July issue of PCW 1 read with considerable interest but tinged with an equal degree of dismay. Mr. Goadby's description of the 2708's structure and its programming requirements were clear and logical as indeed was his description of his 6800 driven programmer. My dismay centres around the design concept of the programmer since it seems to me that Mr. Goadby has missed one of the main reasons for utilising a microprocessor approach — that of minimising peripheral circuitry.

Specifically EPROM addresses are supplied by counters and the required program pulse generated with a monostable. Both these may be eliminated by utilising software generated address lines and program pulse. Indeed software control of all lines enhances overall flexibility and hence power. For example using this approach the programming of selected data blocks, or single locations only, becomes a simple matter. It also provides additional features such as automatic checking for a 'clean' EPROM prior to programming and subsequent verification of correct data.

About 18 months ago I designed a 2708 programmer driven from an Intel SDK80 board and consisting of 1 socket, 6 transistors, 7 resistors and 2 capacitors only; all control lines being under software control and providing the facilities described above. I would be quite happy to supply circuit details to anyone interested on receipt of an SAE.

Dr. A.G. Cartwright.,
University of Surrey
Dept. of Mechanical Engineering,
Guildford, Surrey GU2 5XH.

The EUROPA BUS

I am writing in connection with the proposed E78 Eurocard connector standard published in your magazine recently.

A very short list of some of my criticisms from a professional standpoint are as follows (these are not by any means exhaustive).

1. No attempt was made to contact any of the present manufacturers who have already produced eurocard boards to their own standard.
2. There was no mention of the problems which will occur with such microprocessors as the 8085 and 8086 which have multiplexed data and address busses.
3. The negative supply rails should not go next to logic lines since if a short did occur, the TTL and MOS circuits are more easily damaged by negative voltages.
4. No allowance was made for other interrupt lines such as FIRQ on the 6809.
5. No mention was made as to which control line of WR and RD should be used for the single function of R/W as on the 6800, 65xx and 9900.
6. No mention was made for alternative power distribution systems which employ a regulator on each card to distribute the heat down the chassis and improve noise isolation between boards.
7. No allowance was made for a power supply 'sense' to the far end of the mother board for improved regulation.
8. The row b of the connector should not be connected to row c because of its usefulness in accommodating Non-Standard signals.
9. The single-size eurocard should be recommended since most manufacturers find it quite large enough for their application and future integration can only support this view.

I would very much like to see a pin allocation standard produced, but a much more professional approach. In fact, several of my colleagues and I have already laid the groundwork for such a proposal which will be produced from the collected preferences and suggestions from a great deal of professional engineers, and after approval from all the actively interested parties an approach will be made to the British Standards Institution and the International Electrotechnical Committee for approval and adoption.

I urge all potential users of the E78 standard to think carefully and keep closely in touch with professional circles who are likely to settle on a more useful standard shortly.

I would welcome interested readers to write to me as soon as possible giving their views. I invite the authors of the original article also to submit more detailed reasons on why they picked the various pins for their functions.

Paul L. Borrill.
Department of Physics and Astronomy,
University College London
Mullard Space Science Laboratory,
Holmsbury St. Mary, Dorking, Surrey.

Being 'into' computers yourselves must occasionally cloud your judgement of how dumb it was once possible to be before everything started clicking into place . . .

I have just read Martin Healey's 'MINICOMPUTERS AND MICROPROCESSORS' for the second time; the first reading was of a brand-new copy from Solihull Library, the second time from my own copy — I found it that informative that I could'nt resist spending all of £6.45 for a copy, and that isn't me at all!

Next question is: how do I learn programming; if it is to be from a book in the first instance: what is the outstanding book on (for instance) BASIC, and why is it the best for a raw novice? After reading the July issue Book Review I went into Birmingham and looked-up 'Illustrating BASIC' by Donald Alcock, but immediately formed the view that it — like the other seven books they had in stock on learning BASIC — was more suited to converting someone to the language after having had some previous knowledge or experience of high-level languages.

Consider for a moment the necessities for a 'home-computer explosion': immediate expansion among those who are already computer-trained; further sales to those who have problems which they know can be computer-solved; additional purchases by the curious and by dilettantes; finally: reliance upon expanding sales to the masses. This last step is a pretty rocky one — unless the industry and its' media help matters along.

B.A. Martin
99, Northdown Road,
SOLIHULL,
West Midlands, B91 3ND.

(PCW BASIC is easy to learn and the best way of learning it is to sit down in front of a computer and *try*. Just reading books only gives second hand knowledge).

With reference to the correspondence about Pontoon, Mr. Greenwood's coding would not fit into the program as written, nor would it shuffle the numbers 1, 1, 1, 1, 2, 2, . . ., 12, 12, 13, 13, 13, 13. A coding which would work and could be inserted into the program as written is given below:—

```
 60  FOR A = 1 TO 52
 70  LET P(A) = INT ( (A – 1)/4) + 1
 80  NEXT A
 90  REM PACK NOW CONTAINS 1, 1, 1, 1, ETC
100  FOR A = 52 TO 1 STEP – 1
110  LET R = INT (RND (1) * 52 + 1)
120  LET V = P(A)
130  LET P(A) = P(R)
140  LET P(R) = V
150  NEXT A
```

I would also like to suggest that line 780 should be changed to:—

```
780  GOTO 50
```

which avoids looping through the line 40 RANDOMIZE, as this can cause problems in some BASIC dialects.

W.S. Lounds
65, Blades Street,
Lancaster.
LA1 1TS

ERROR MESSAGE

A couple of 'Bugs' crept into my 2708 programmer:

1. C2 = 1000 pf (Fairly obvious)
   R2 = 27K
2. TRANSISTORS ARE PREFIXED BY 2 NOT BY Z.
3. PIN 13 of 1C6 should go to the VERIFY contact of 51. A 1K resistor should be connected between this contact and + 5 volts.

David Goadby,
2 Lupin Close,
Hinckley,
Leics. LE10 2UJ

A Blast at Poor Service

This is the sort of letter that asks the Guilty to stand up and be counted. After 8 years of small computers research, I recently bought myself a Tandy TRS 80, and in one week I became a personal computer owner with a bang, as with my TRS 80 I was carted off to the BBC TV studios to appear on 1, and on radios 2 and 4 to give demos. Well this naturally created a great deal of letters, from all over the country asking for help, which I am pleased to give.

But one strong fact came up time and time again, and it happened to me, and that is there are dealers, agents, and manufacturers that give a poor service in the way of fast promises, but slow deliveries of equipment, information, and hard or software. One well known software shop states 24 hour service, in fact 24 days is the case if you're lucky. I have read that many people write to many companies that I know well, asking for help, as they want to compare notes before they part with their hard earned cash, and they are still waiting. Now, I think that this is disgraceful at a time like this when the computer boom is just starting in this country, as it is good business for everybody. Yet we the cash holders are slapped down by the British companies' attitude of "you are a small man, you'll wait". So stand up and be counted those that do not give good service.

Well, a lot are standing.

P. Turner, BSc., M.R.I.M., M.I.E.D.
14 York Way, Thetford, Norfolk,
IP24 1EH

Are imported computers really that overpriced?

May I refer to the letter from Mr. Kit Spencer, the General Manager of CBM, published under "Error Messages" on page 76 of the July edition.

This company imports scientific equipment from the USA and distributes it in the British Isles, and therefore we could easily have a PET computer purchased for us in the States and shipped to us. As someone who really knows the costs involved in importing, I can assure you and your readers that, when we are ready, if we decide that the PET computer is the best one for our needs, we shall be purchasing it *in* the UK and not asking our American colleagues to obtain it for us.

Alan Cussens,
Managing Director,
Bristol Industrial & Research Associates Ltd.
P.O. Box 2, Portishead, BRISTOL BS20 9JB.

I disagree with D.J. Chown's objections to the term "personal computer". Thanks to the often twisted dramatic representations of computers and their close relatives the robots, the average person has a fully incorrect picture of the capabilities of computers. The personal computer can do much to remove the mysticism from programming and computers. The argument that the "personal" designation gives the Joneses their usual one-upmanship feeling in relation to the Smiths and Millers, who only have a 'home' computer is really a sad recognition of the extent to which all of us are influenced by social snobbery. Whether we have a 'home' computer, a 'personal' computer or allow ourselves to be designated as 'hobby computerists' (what a beautiful way of degrading someone to the lowly ranks of cranks, eccentrics and knob twiddlers), the important fact is that the man in the street gets closer to the realities of the world of computers and the so-called 'thinking' machines, and begins to realize that the 'thinking' has to be done by man.

My concern with personal computers is that the amateur computer user is being exploited by companies who are so spoilt by the industrial markets that they squeeze every penny they can get out of the purchaser. The European retailers are the worst examples of this extortion. The prices asked for equipment manufactured in the USA and sold in Europe are nearly always around 200% of the US price. I can appreciate that a certain overhead has been incurred for freight, but from experience I know that these costs are only in the order of 25% of the US price. The remaining equipment often has little or no maintenance liability attached, and assistance with installation and bring-up is virtually non-existent.

I feel that PCW should start a campaign for fair pricing by publishing facts on country-of-origin costs, freight costs etc. and by promoting a PCW booby prize for the most exorbitant profit margin of the month.

A big step in the right direction could be made by only accepting advertising with precise pricing information. Some companies advertise with 'from (price) . . . excl. VAT' but when you get down to details, the quoted price is for a non-viable system, and to arrive at a *viable* system more than double the 'from' price is required.

As comment to PCW itself, I think we can do without satirical (?) articles of the type 'A day in the life of . . .'. The space in our magazine is too precious to waste. This space could be far better filled with a 'Where can I obtain it' section which lists accumulated information on addresses where the various items can be purchased, the format could be similar to the 'yellow pages' concept. The beginner and the old hand both have a need for such information, especially since the market is expanding so rapidly. I'm sure your readers would be only too glad to contribute to such a list, not to mention the suppliers themselves who would profit by such a free listing. If you do start such a list please do not limit it to the UK!

P.J. Devereux.
7036 Schoenaich
Langobardenweg 13
W. Germany.

Tid Bits PRODUCTS . . . . COMPANY NEWS . .

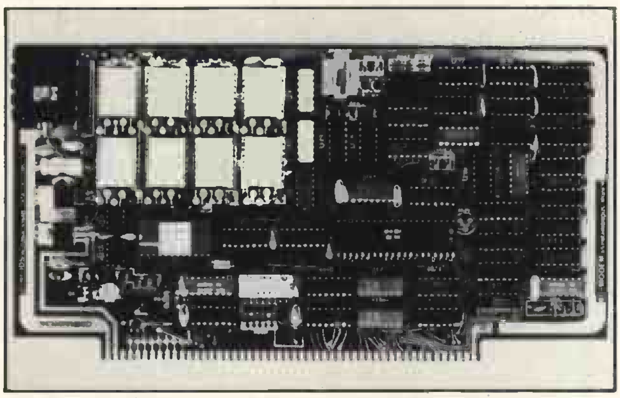
THE MODEL 88-MODEM – MODEM MODULE FROM INTERNATIONAL DATA SYSTEMS, INC. provides communications over either the switched telephone network or private lines at any software selected baud rates between 66 and 600 baud. The 88-MODEM is fully compatible with Bell System type 130A MODEMS and provides either half or full duplex operation. The 88-MODEM is S-100 bus compatible and includes a serial I/O port and an originate/answer MODEM on one board. All features are implemented in hardware including pulse code dialing in originate mode, automatic break/disconnect, and dial-tone detect.

The 88-MODEM includes an 8-Pole transmit and 8-Pole receive filter, self-test circuitry, dial-tone detect filter as well as standard error detection circuitry including parity, overrun, etc. The dial-tone detect circuit allows the dialtone to be positively identified prior to auto-dialing or originate calls.

Extensive software is included with the 88-MODEM including both stand-alone and handler originate, answer and patch programs. Patch programs are currently provided for MITS BASIC and the North Star DOS version 3. These routines allow remote access of S-100 systems via dial-up computer terminals.

The 88-MODEM is fully compliant with all applicable Bell System tariffs as well as Part 68 of the Federal Communications Commission rules and regulations.

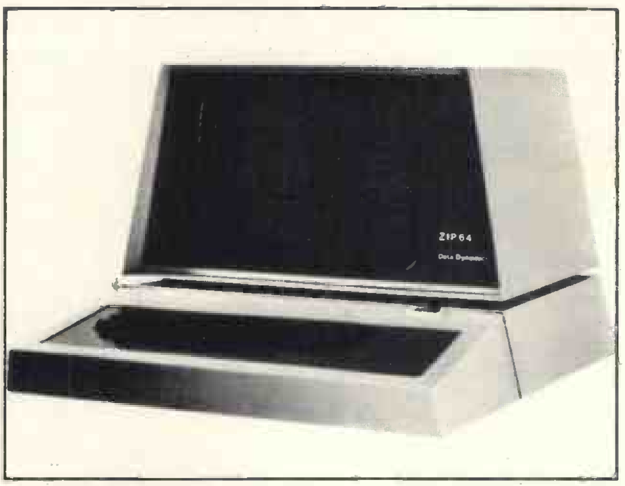
The 88-MODEM is available in kit or assembled form from International Data Systems, Inc. at 400 North Washington Street, Suite 200, Falls Church, Virginia 22046 or telephone (703) 536-7373.



RAIR of 30/32 Neal St. London WC2H 9PS, is moving into the expanding field of microcomputers with its new RAIR BLACK BOX Microcomputer.

The BLACK BOX is a fully integrated floppy-disc microcomputer featuring the latest Intel 8085 microprocessor, integral single or dual minifloppy disk drives, dual serial I/O ports, housed in a self-contained desk-top cabinet. Software includes a powerful floppy-disc operating system, advanced BASIC Interpreter, and FORTRAN and COBOL compilers.

RAIR sees the microcomputer industry as having a dramatic impact on the computer market. With complete systems available under £2,000, few businesses will be unable to take advantage of the power of their own stand-alone microcomputer; and will have the opportunity to consider many new applications for cost effective computerisation.



The ZIP-64 Visual Display Unit from DATA DYNAMICS has, according to the Company, the following features:

* Very low cost
* 64 characters per line
* 16 line display
* 110-1200 baud
* Video output for two monitors
* Datel V24 (RS232C) and 20mA Interface

ZIP-64 is a general-purpose visual display unit which employs the very latest in semiconductor large scale integration to achieve very low cost without sacrificing performance. The 12-inch display tube accommodates 16 lines of 64 characters in 5 x 7 dot matrix format. The standard 64 subset of ASCII is displayed providing upper case letters, numerics and punctuation marks.

User selectable facilities include parity, full/half duplex, 1 or 2 stop bits and baud rate (110 to 1200 baud). Interface is CCITT V24 (EIA RS232C) or 20mA current loop. A video output is provided, capable of driving an external video monitor. The monitor should have a standard 75Ω /1Vp-p input and, if the equipment is to be used on British Post Office lines, only a Post Office approved monitor should be used.

ZIP-64 is a member of the Data Dynamics ZIP family of communications and computer terminals which include ASR, KSR and KDP (keyboard, visual display, printer) units.

Microsystem Services Publish PROM Programming Brochure

Microsystem Services have published a brochure which will be of interest to all engineers concerned with PROM programming.

The brochure introduces the range of PROM programmers manufactured by Data I/O and supplied in the UK by Microsystem Services. A comprehensive list of PROMs from twenty manufacturers is included and provides such details as memory size, technology, programmed logic level and number of pins. The list also gives the part number of the program card set, socket adapter and other equipment needed to program each PROM type.

A page is devoted to describing the functions performed by the items of equipment required to program PROMs.

In addition to specializing in PROM programmers, Microsystem Services also supply 8080A microcomputer development systems from muPro, military specification Nova compatible computers from the Rolm Corporation, and an interesting forced cooling system for testing components manufactured by Termonics.

Further Information:
Jim Knott,
Microsystem Services
Duke Street,
High Wycombe, Bucks
Telephone: (0494) 41661

Intel's 8049 is "Now the World's Fastest Single-Chip Microcomputer"

Intel's top-of-the-range single-chip microcomputer – the 8049 – has been upgraded to run at 11MHz instead of 6MHz as originally specified. This dramatic 80% increase in speed, which enables the 8049 to perform 16-digit decimal addition in only 96.7 μsecs, has been achieved with only a 30mA increase in power supply current – from 140mA to 170mA.

The 8049, now the world's fastest single-chip microcomputer, can be employed with the very low-cost 10.7MHz ceramic filters which are mass produced for f.m. broadcast receivers.

The 8049 is an 8-bit parallel microcomputer with 2K bytes of program memory, 128 bytes of read/write memory, an interval timer/event counter and 27 I/O lines fabricated on a single chip.

The 8049 has over ninety instructions (70% of which are single byte), all of which are executed in either one or two cycles. The instruction set has been optimized for control applications and for arithmetic processing. It contains efficient table look-up instructions and single instruction 'test and jump' instructions as well as extensive facilities for bit manipulation and binary and BCD arithmetic.

The older 6MHz 8049 is no longer available and all 8049s are now capable of 11MHz operation. The 8039 (functionally identical to the 8049 but with no program memory) has also been upgraded to 11MHz. However, a 6MHz version, the 8039-6, will be available.

Further information:
Brian Crank
Brian Crank Associates
Annery House
53b Frant Road
Tunbridge Wells, Kent
Tel: (0892) 31812 and 38414

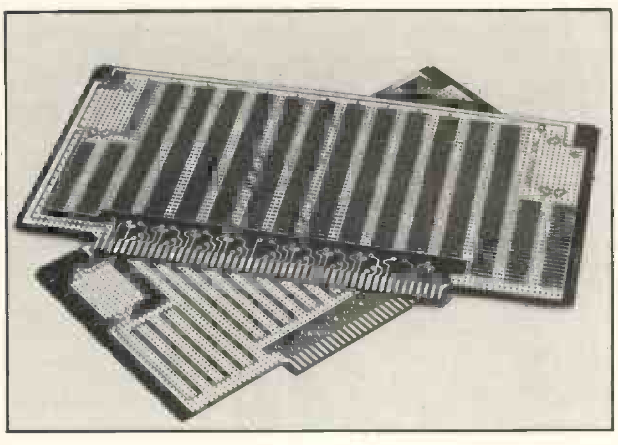
John H. Miller-Kirkpatrick of **Bywood Electronics** has written two little books for the beginner in computing: "Microsense 1" and "Microsense 2". The style is engaging and clear. The booklets are invaluable for understanding SC/MP mpu based systems; in particular the Miller-Kirkpatrick designed "SCRUMPI" series of computers. Booklets (£2 +20p P&P for the pair) obtainable from: Bywood Electronics, 68 Ebberns Road., Hemel Hempstead, Herts. Phone: 0442-62757

S100 Universal Microprocessor/Microcomputer prototyping board

Following the increasing use of the S100 board size and bussing system in microcomputers (e.g. Altair 8800, IMSAI 8080) and a microprocessor applications, Vero Electronics Limited announce the release of a universal S100 bus-compatible prototyping board. This board is designed for the manufacture or breadboarding of microprocessor, memory or interface assemblies, and will, without modification, mount directly into any equipment using the S100 bus system.

The layout of the Vero S100 prototyping board has been optimised for maximum flexibility in use and as a memory board will hold up to fifty-two 16-way DIP's (equivalent to 6K of memory) or in more general use, thirty-six 16-way plus eight 24-way plus two 40-way packages, making it ideal for microcomputer expansion and general digital and analogue circuits.

The board has an S100 edge connector configuration (i.e. 100 gold-plated contact fingers on 3.175 mm/0.125 inch pitch) and is fully pierced with 1.02 mm/0.040 inch diameter holes on a 2.54 mm/0.1 inch matrix. Provision is made for mounting up to four standard TO-220 plastic package regulators together with heatsinks for on board regulation, and the voltage plane is capable of being divided to provide up to four separate positive or negative supply rails. The comment side of the board carries a ground plane which can be used for terminations or screening and the wiring side carries both voltage and ground planes, thus providing for up to five planes.



Vero Electronics Limited,
Industrial Estate, Chandler's Ford,
Eastleight, Hampshire. SO5 3ZR
Telephone: (042 15) 69911 Telex: 47551
Contact: Alan Young Reference: S100 AB 043

Online is presenting a seminar on Computer graphics from 12–15 September. Main attractions are American experts *Carl Machover and Bert Hezog* but *Derrick Grover* of NRDC (a Computer Aid design specialist) got Britain into the act and will speak on September 15th, bringing (the organisers say) the event to "a fighting finish". Event is being held at the Regent Centre Hotel, London. Details from: Eddie Dawe, Online Conferences Ltd., Cleveland Road, Uxbridge, Middlesex. Tel: (0895) 39262.

PCC SIGNS AGREEMENT

PERTEC COMPUTER Corporation (PCC) has signed an exclusive wholesale agency agreement with **Compelec Electronics Ltd**, London, for the distribution of small business systems in the United Kingdom and Ireland, Donald F Orr, PCC vice-president and general manager of the Business Systems Division, announced.

"Under the terms of the agreement, Compelec will expand its activities support of the MITS product line, which is sold in Great Britain using the Altair brand name," said Mr Orr.

Mello Van Reigersberg Versluys, chairman of Compelec, said that the new, formalised agreement making his firm the first MITS distributor in Europe would enable Compelec to establish a network of dealers and direct sales outlets in the United Kingdom and Ireland.

Additionally, he said, Compelec intended to work directly with some 50 established software firms throughout Great Britain to produce applications programs for MITS/Altair systems.

Further Information:
Carlos Versluys, Compelec Electronics Ltd.,
107 Kilburn Square, Kilburn High Rd., London NW6 6PS.

The National Computing Centre (NCC) has just released its annual report and accounts. It reports a successful year ended March 1978, with a surplus of £149,068 (pre-tax). The previous year showed a deficit of £53,725.

The report indicates the vitality of the NCC.

The NCC is actively engaged in studying the impact of new technologies on the economy and society and is playing a significant role in training and education. It runs an elightened – and successful–training scheme for school leavers. This "Threshold Scheme" is funded by the Training Services Agency. Over 400 trainees entered the Threshold Scheme, and 86% of those who completed training found employment in data processing.

The NCC is concerned about producing and implementing standards, and has produced a report on this for the Department of Industry. Its activities are wide ranging and include joining in European programmes of work on Database, Privacy and Security, and Programming Techniques.

The NCC has jointly with French and German groups published GINDEX, a glossary of Computing terms. No Computer professional should be without this glossary.

Further information:
NCC, Oxford Road., Manchester M17 ED
Telephone: 061-228 6333

First ever International Computer Retailers' Conference

"Opportunity for sales and profits in today's explosive computer retailing market".

This is the theme of the first ever International Computer Retailers' Conference to be held in Chicago, October 25-27. It is being organized by Management Research Associates, conference and exhibition organizers of New York City, and sponsored by Computer Retailing, Geyer's Dealer Topics and AudioVideo International.

Gene Wolfe, President, Management Research Associates, states: "A new revolution in retailing that began three years ago with the opening in California of the first computer store is now rapidly gaining momentum and is spreading like a prairie fire to all kinds of shops and stores. It won't be long before thousands of department stores, radio and TV, camera, hobby, audio video, office equipment, electronic part stores, etc. will be stocking microcomputers in New York, Los Angeles, San Francisco, Pairs, London, Tokyo, and Frankfurt.

"How to sell microcomputers to the small business, hobby, educational and home markets is a hot subject with a growing number of retailers. The main purpose of this conference and show is to provide existing and future computer dealers with an in depth look at the opportunities and, not least, the pitfalls for developing sales and profits in this expanding area.

"America is now where the action is in computer retailing, but it is destined to become big business with retailers throughout the world. By 1980 it is predicted that personal computers will average $ 300. With prices down at this level it opens up a huge consumer market for the retailers".

Registrations for the "International Computer Retailers' Conference and Show," restricted to retailers, computer manufacturers and distributors, should be made to: Registration Manager, Management Research Associates, 60 East 42nd Street, New York, N.Y. 10017 (212) 687-2560

TOPMARK Computers have been appointed sales agents for East Anglia and the North Home Counties for the APPLE II personal computer.

APPLE II features light weight and colour graphics and is the ultimate personal computer. Applications include process control, data logging, design, statistical analysis, accounting, management statistics, automatic testing, laboratory research, teaching, audio-visual display, numerical control, microprocessor development, as well as games and problems.

APPLE II includes full keyboard, 8 I/O ports, 1000 cps display, BASIC and Monitor in 8K ROM, up to 48K RAM, fast cassette interface (1500 bps), analogue and TTL inputs, and speaker.

BASIC includes any length variable names (ALPHA, BETA$), syntax and range errors indicated as soon as entered, multiple statements on one line, string arrays, mixed text and graphics (software selectable), auto line number, direct memory access, and line number and variable trace for debugging.

Powerful Monitor routines include dis-assembler and mini-assembler, multiple commands on same line, examine, change, move, and verify memory, examine and modify registers, set for inverse or normal display, hex add/subtract for relative branch calculations, and single step and trace modes.

A wide range of options is available. These include voice operation (commands and data can be stored on cassette in voice form for later analysis), floppy disc, teletype and printer drives, and high resolution graphics.

Full details from Tom Piercy at TOPMARK Computers, 77 Wilkinson Close, Eaton Socon, Huntingdon, Cambs, PE19 3HJ. Phone Huntingdon (0480) 212563.

ROCKWELL'S NEW LOW-COST MICROCOMPUTER HAS ALPHANUMERIC PRINTER, DISPLAY AND TERMINAL-STYLE KEYBOARD

Rockwell is implementing a long term product development programme stemming from the R6500 NMOS family as part of a "total microelectronic system solution commitment", according to Bob Anslow, director LSI Products at Rockwell's Microelectronic Devices; and Gordon Dale-Smith of Pelco is "looking forward to introducing a stream of new products that epitomise Rockwell's advanced technological capability over the next two years".

Hot on the heels of the single chip 6500 microcomputer designated R6500/1, which was announced last month, comes a new single board, low-cost microcomputer system featuring an on-board 20 column printer and full alphanumeric keyboard.

Designated AIM 65 (R6500 Advanced Interface Module) the new system is intended as an educational aid for first time users and a general-purpose microcomputer for engineers. AIM 65 is also designed to be an effective, low-cost microcomputer development system priced at under £250.00.

AIM 65 features on-board alphanumeric 20 character printer and display and a 54-key terminal-style keyboard. Available in 1K – and 4K-byte RAM versions, AIM 65 is designed around the 6502 CPU, which has 64K address capability with 13 addressing modes, and is the microprocessor at the heart of other popular systems such as KIM-1, PET and APPLE.

An 8K ROM-resident monitor programme provides all peripheral control and user programming functions. Spare sockets are included to further expand on-board programme memory via user PROM-based programmes or Rockwell's assembler, text editor and BASIC interpreter plug-in ROM options.

The AIM 65 board also has a connector that allows external access to the system bus for memory and I/O expansion. A separate application connector interfaces a teletype and two standard cassette recorders, and includes a user-dedicated Versatile Interface Adaptor. The VIA features three 8-bit bidirectional ports (two parallel, one serial) and two 16-bit interval timer/event counters, thus allowing the user to interface his own system, in many cases without the need for extra interface devices.

The product is being stocked by Pelco (Electronics) Ltd., Enterprise House, 83/85 Western Road, Hove, Sussex, BN3 1JB, (telephone: 0273-722155), Rockwell's U.K. Distributors and Representatives.

I BYTE

BUT COMPUTERS DON'T

Come and see for yourself at

the BYTE SHOP Ltd

Stockists of the largest range of micro computers in the U.K.

Take the opportunity to experiment with and get to know any of the vast range of micro computers always in stock at The Byte Shop.

Whether you want a micro computer for your home, your business, for industry, for education - or if you'd just like to find out which model you get on with best - you'll find a visit to The Byte Shop a new and invaluable experience.

Call in at The Byte Shop any time from Monday to Saturday. It's right by Gants Hill tube station.

The Byte Shop 426/428 Cranbrook Rd., Gants Hill, Ilford, Essex. Telex 897311
Telephone 01-554 2177

SO YOU'RE INTERESTED IN PERSONAL COMPUTING

Brian Hewart

Your first look into a computer magazine is just like picking up a sheet of music for the first time. In your mind you know what it sounds like but on paper it's just a lot of dots. Now sit in front of an organ with a keyboard guide and bit by bit, one finger at a time, the dots start to make sense. It's the same with a computer — play with one for a few days and suddenly it all falls into place.

So the first advice that I can give you is beg, borrow, build or buy a computer and stop wasting time reading about them.

However, you've read this far, so let's give you a mini guide to personal computers. First, we divide a computer into two parts: Hardware and Software (doesn't the electronics industry love making up names to confuse people).

Hardware

This covers everything that, when dropped on your foot makes you say ever so softly, "Oh dear, what a silly thing to do!" In other words it's the solid electronics of the thing that you would normally identify as a computer.

Software

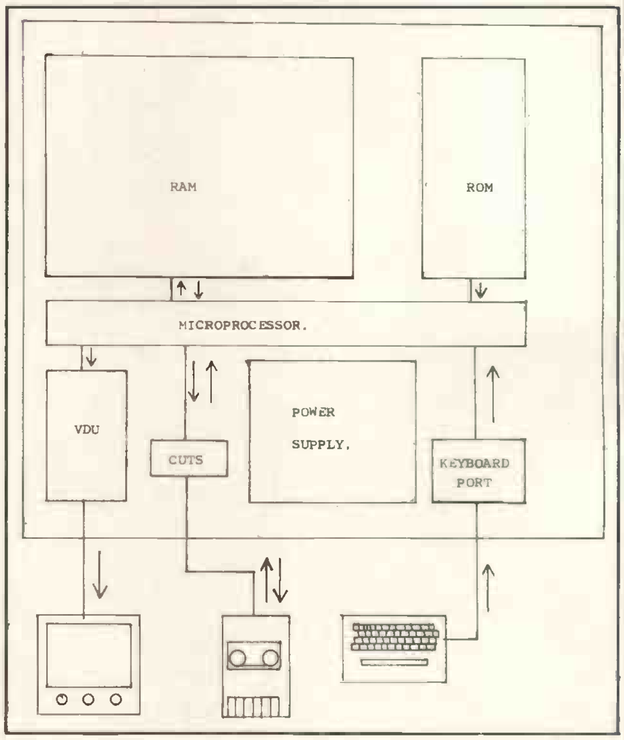
This covers the hidden programs inside the computer. The side you don't normally see: but it is this part that makes a computer so attractive to the enthusiast. If you feed in a 'Mastermind' program the computer is a game (and I can vouch for this one as being a very entertaining one for all ages). You can then feed in a program to do a stock check of your freezer (vitally important when you have to convince your wife how much you need a computer). And so on, you create whatever system you want simply by changing the software. Unlike popular video games units, a computer can never become obsolete, so whenever choosing a system get one that can be expanded to cater for your future needs.

Lets Take a Deeper Look Into Hardware

The first hurdle you have to cross is BITS. This is short for Binary Digit. In a computer all data is in binary, that is either 0 or 1, and it is usual to have a 1 represented as a high voltage and an 0 as low voltage. To make up bigger numbers we use a number of wires carrying high and low voltages (0s and 1s) in parallel. Hence you will see 4, 8, 12, 16 and 32 bit computers written about. Nearly all amateur computing is being done with 8 bit computers, so for simplicity forget the rest. When we have 8 bits in parallel this is called one BYTE. You may see the statement 'an 8 bit word is one byte'. A word can be any number of bits that has some particular function in a computer.

Now that we have sorted out the bits and the bytes we can move on to the MICROPROCESSOR. This is the heart of the system. In no way is it a brain. It does one job only, it carries out your instructions. It is best thought of as a willing idiot, it will do whatever you tell it to (within it's capabilities) so that if you make a mistake in your program — hard luck, you've blown your program. Say ever so politely to yourself "Oh, how silly I am", and start again. Everybody makes mistakes when writing programs, don't give up if your first try doesn't work.

What is the difference between the available 8 bit microprocessors? On the whole, to an amateur, not a lot. They will all do roughly the same job but the better chips (short for integrated circuits) require fewer instructions for the same job. The amateur computer scene at the moment seems to be centred on the Z80, 6800, 6502 and the SCMP chips. The Z80 is



Typical home computer showing interconnections, signal directions and relative size of the sections of the computer. Note how all the signal paths go through the microprocessor.

tops technically but maybe a little complicated to program. The 6800 and 6502 are very similar, very popular chips. The 6502 is used in PET but is not so readily available in the country. The 6800 is readily available and very easy to use and program (yes, you've guessed — I am a 6800 man). Last is the National SCMP which for some reason has always been looked down on but still has a strong following. Possibly National created the wrong image by promoting it as a Small Cost Microprocessor.

To an amateur the most important thing is to have programs to run on his machine. On this score the 6800 and Z80 stand out as having mountains of programs available — just look at the adverts. Also magazines such as PCW publish programs regularly.

There are better 8 bit processors on the horizon but don't worry about them, and certainly don't put off getting started because of them. It is in the nature of things that there will always be better chips on the horizon; but they cannot make the existing ones obsolete when they appear.

We now have an electronics block chewing data 8 bits at a time. Where does this data come from? Three main sources: External input, RAM, ROM.

External Input. Data is fed into the system from the outside world. This can be from a keyboard, teleprinter, switches, tape recorder etc. The entry point on the printed circuit is called, believe it or not, a PORT. This can be a parallel port where 8 bits are fed in via 8 wires (you can have more or less bits), or a serial port that has only one wire and the bits are sent in one after another.

RAM. Short for Random Access Memory. This is temporary storage that can be pictured as a vast wall of pigeon holes, each numbered, in which we can store and retrieve data at random. Since we are using an 8 bit system each hole should store one byte. How many bytes do we need? There is no answer to this, the more the better, but I would say 1000 (1K) bytes is reasonable starting point, enough to learn programming and run some good programs. You do not see a computer acting like a computer until you have enough memory to store a BASIC program. This takes from 4K upwards. Basic is a language that lets you enter instructions in specialised English whereas you have to use 'MACHINE CODE' otherwise. Machine code is simply a set of numbers that the computer understands and programming in numbers is harder unless you can remember several hundred numbers.

It is a strange fact but the bulk of the cost of a large amateur computer is in the memory.

ROM. Read Only Memory. When you switch off power to RAM it loses its data. When you switch on again you have to have some data to tell the computer to get new data from a keyboard or tape recorder.

We store this data in Read Only Memory because this type stores its' data in a way that needs no power. You cannot normally enter data into it without special equipment, hence 'read only'. You can have the simplest possible data in ROM and always enter the main operating program from tape (very good but slow); or you can have a bigger ROM and have a quite complex 'monitor' program always ready when you turn on. A further extension is PET that has a full BASIC in ROM so that it is always available (this is probably the ultimate for the amateur who does not know which end of a soldering iron to pick up).

How does this data get travel round the computer? By bus of course.

A bus is the term given to any length of wire that has wires joined onto it at various points. Your house has it's power fed from a bus under the road.

In a computer there are normally three busses. The **data** bus, which in this case we are assuming is 8 bits wide, consists of 8 wires or printed circuit tracks that thread their way to all points that require data. That is, the processor, RAM, ROM and input, at least. The data bus is bi-directional, that is data can flow into and out of anything it is connected to (ROM excepted of course). The direction of data is controlled by the processor in the form of a READ WRITE line (usually called read/not write) which means that if the line is logic one the processor is reading from somewhere, if the line is zero then the processor is writing to (storing data) somewhere. You will see this 'not' cropping up a lot. It is also shown as a bar across the top of some data such as write above. What it means is that the statement with a bar on top is true or active when the data is *zero* voltage. A common one is *not* enable (enable) which means that whatever is being enabled is enabled when the control line is at logic zero. Another phrase for it is inverted logic. If you look at a properly drawn circuit diagram you will see the active low inputs shown with a small circle. Similarly inverted outputs come out through a small circle.

The **address bus** normally is 16 bits wide (double the data bus). When the processor wants to store or retrieve data it has to say where this is from/to. It does this by outputting an address on the address bus. If you work it out you will find that there are 65,536 possible combinations of the 16 address bit. This is where we see that the processor is capable of addressing 64K of memory. Just to confuse you it has become common practice to call 1024 bits, 1K so the 65,536 is wrongly called 64K except by those salesmen who want their product to look whiter than brand X.

An ADDRESS DECODER, which is a set of gates wired up to respond only to one particular address or set of addresses is connected to the address bus and to the part of the computer being controlled so that, that part is turned on only when 'addressed'. In other words 'speak only when you're spoken to'.

The last bus is more of a minicab, this is the **control bus**. In the 6800 it can be as little as three wires: The read/write mentioned above, a signal called V.M.A. (valid memory address) and phase two of the clock. The system clock is only a high frequency oscillator that sets the speed at which the processor works.

Well, that's the transport system of the computer and if you think about it, it is also one of the least liked parts of it because you have about 30 wires to connect when joining printed circuits together.

Now we have an 8 bit processor chewing data from RAM, ROM and Input. What do we do with it? One of two things. We make the data work, such as turning on the central heating, TV, lights, security system. Or more often we simply take the data into our brain. How? Two choices again — sound and sight. There are available, voice synthesisers that will let the computer talk to you. What a fantastic improvement these will make to the life of blind people; but for the rest of us they are a little expensive at the moment. The more usual visual output from the computer takes three forms: the calculator style display of varying size, the teleprinter and the television display.

You will never approach the wonderful possibilities of even a 1K computer with a few numbers on a calculator display, so save your pennies and wait until you can afford a better computer. All the early computers used teleprinters. These are very useful but very expensive and use vast quantities of expensive paper. The last output device is the British national pastime, the TV. For most computing the TV display is perfect and cheap.

To convert the numbers that the computer spits out to a suitable signal for a TV we have the VDU or Video Display Unit circuit. These take many forms depending on cost. The average one will display 1024 characters on the screen in a 64 column by 16 row format. Again the average VDU will output 64 characters consisting of capital letters, numbers and punctuation marks. The more expensive ones generate small letters and shapes used for building up pictures (so called Graphics capability). There is not yet much software that generates graphics so this is not essential. The American systems generate full colour (they've not yet perfected colour) but we poor British can't afford this yet. Whenever you think of buying a system with 'frills', don't. Buy a good basic system and save your spare cash for more memory. Don't spend £40 on a flashy case, use an old hat box and buy another 4K of RAM.

This completes the brief tour of the Hardware – you feed data in – you get data out, the in between is controlled by Software.

Software. This is nothing more than a set of numbers, but because it takes time to put it together and debug it, software can be quite expensive. The popular stuff, sold in large quantities, is really very cheap and a lot is included in the better magazines such as this one.

To show you what it looks like here is the smallest working program that you are likely to see. It calculates the relative address offset used in programming, or in English it subtracts two numbers. It is written for the 6800 microprocessor.

0000	86 05	Start	LDA	Load A 'TO'
0002	80 04		SUB	Sub A 'From'
0004	B7 00 0A		STA	Store A 000A
0007	7E EO E3	End	JMP	Jump Control

First Line. Starts with RAM address 0000, the very first location. Here we store the command 86. This tells the 6800 to load into accumulator A (a RAM inside the 6800) the data stored in the *next* memory location. 05 is used as an example.

The computer now moves to the next RAM location expecting another instruction. At 0002 it finds 80 which tells it to subtract A (now holding 05) the contents of the next memory location which is 04. Now acc.A holds 05 – 04 = 01.

Next instruction is B7, this tells the 6800 to store A at the memory specified in the next TWO memory locations, which is 000A.

It moves on and at 0007 gets the command 7E which is the command to *jump*. It jumps to address E0E3, which, for a 6800 system with Motorola Mikbug as its' control ROM, is the signal for the end of a program. You would now examine the contents of 000A and find the answer to the calculation. Once you had started the program it took the 6800 about 15 microseconds to perform it. Suppose you made a mistake and at address 0004 you put B7 0009. This would have stored the answer where E3 is in the last instruction so the computer would have jumped to E001 and that is the last we would see of a sensible program. You would have to press the reset to regain control.

This is what software is all about. To give you an idea of how many instructions are needed to coax your willing idiot through a real program, Mastermind (or Bulls and Cows etc.) takes about 800 bytes of program. Noughts and crosses takes about the same (very boring game because the computer cannot lose.) A lot of simple number games take 200/400 bytes. Star Trek is a very popular game at 4K bytes, but games like chess run into tens of thousands of bytes.

Precisely Speaking

Sheridan Williams

To the scientist and engineer the computer is just a tool, it is an instrument just like a calculator, a slide-rule, or a micrometer. The computer user must be able to trust the output that the computer gives. Apart from errors which arise from mistakes in the program, there are errors that arise from the computer itself. We have all been told at some time or other that computers do not make mistakes, this is true, but they do have certain inaccuracies which creep into the calculations, It is essential that the programmer is aware of where these inaccuracies may occur and know how to minimise them. The whole topic of errors and they ways in which they may be predicted and minimised is covered in "Numerical Analysis". This is a topic which is studied by all students of computing. (Indeed, if it is not, then it should be).

The importance of Numerical Analysis in computing is twofold — firstly, because the speed of the computer is phenomenal (by human standards) a computer can do work hundreds of times faster than a person; and secondly computers do not make mistakes, so your results can be trusted. (Provided that your programs have been thoroughly tested with *good* test data). This speed and absolute trust in the results can cause serious problems, and this is why Numerical Analysis is so important.

The scientist and engineer want an accurate as well as a fast tool, and hence they will place their trust in the computer's results. It is up to you as a programmer not to betray that trust and state the answer to 8 significant figures when only 4 figures are correct. It is hoped that you will be guided by this article in some of the techniques of numerical analysis, and maybe motivated to read further; suggested further reading is given at the end.

Before going into detail on computational methods, it is perhaps worthwhile considering why computers were invented in the first place. They were invented for scientific not data processing reasons by Charles Babbage in 1830. He wanted a machine that could be used for making tables (not wooden ones) that were of guaranteed accuracy and error free. He designed two "engines" — the 'Difference engine', and the 'Analytical engine' If *you* had designed them then you would have called them "engines" too. They were large, bulky, mechanical machines. Babbage was Professor of Mathematics at Cambridge, and was an extremely brilliant man; it was a shame that he was not born 100 years later because technology was not sophisticated enough in 1830 to build his machines accurately enough. He died rather a frustrated man. I believe that his brain has been preserved and is on view at the Royal College of Surgeons.

It was the Second World War that provided the impetus for development of computers. This time electronics was available to help. It is worth noting that most of the people involved in their development had never heard of the work of Babbage, and were unaware that their ideas were already 100 years old. There were two reasons this time for the development of computers: firstly to calculate the trajectories of shells and rockets; and secondly to help in the tedious task of codebreaking. Even though computers were invented for scientific reasons, their use now is over 90% in the field of data processing.

To get back to the point, let us examine why we use computers in the scientific field. Although our brains are capable of good logical thought, they are slow and inefficient when it comes to a simple calculation such as 1234.5 x 678.9 ; they are also unlikely to get the answer right consistently. The problem just given may take you a couple of minutes, but in this same space of time a computer would have done over one million such calculations. It would take you around two years working all night and day to perform an equal number of calculattions. That is : a computer can do in 2 minutes what it would take *you* 2 years to do; and the computer would give the correct answer every time — could that be said of you? Ah but wait, are the answers really correct, each answer may be 0.00000001 out, but nevertheless they are wrong. A single inaccuracy of this magnitude may appear insignificant, but after a few thousand calculations involving such errors they may accumulate to a substantial amount.

In order to understand how some of these errors creep in, we need to know something about binary. Binary is not frightening, it is just a different number system than the one that we are used to. Always remember that a computer has no powers other than those we give it; they are simpletons with even less brain power than a lobotomised flea. Their main advantage is that they are infinitely patient, and have the capability to perform a series of tasks flawlessly. Computers use binary because, being based on electricity and magnetism, they are composed of two-state devices; that is devices that can exist in one of only two states. (Example — a switch on or off, a current flowing or not flowing). Binary is a number system that uses two digits only (0 and a 1). Denary (meaning base ten) uses ten digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Certain problems exist when trying to store non-integer numbers inside a computer; these problems arise whenever we are restricted to a finite size of store and are independent of the number system used. The problem would not be alleviated if computers did not use binary. As an example how would you store 1/3 as a decimal exactly? Similar such numbers exist in any number system. In binary the denary number 0.1 is a recurring decimal, it is 0.0001100110011 . . . .
I don't wish to go into detail about binary because it is a simple matter to find out about it from your younger brother or sister, who do "modern maths" at school. I will just convert 110001.11 into denary as an example:

2^5	2^4	2^3	2^2n	2^1	2^0	.	2^{-1}	2^{-2}	2^{-3}
32	16	8	4	2	1	.	½	¼	
1	1	0	0	0	1	.	1	1	

Add up the numbers above the ones and you get the answer: 32 + 16 + 1 + ½ + ¼ = 49.75. Extend the table to the left and right as far as necessary.

So it is in trying to represent numbers like 3.1 in binary that we get stuck — it is not possible. We have to make do and store them as accurately as we can, this usually means between 24 and 60 binary digits (bits) in a computer. This dilemma is in fact one of the many sources of errors in computing.

There are two types of numbers used in computers, they are "integers" and "non-integers" (usually referred to as "real" numbers). Each type has its own arithmetic circuitry, and integer arithmetic is much quicker. (Always use integer variables wherever possible, this will speed up your program). Most real numbers are held in "floating-point" form, in this way both extremely large and extremely small numbers can be represented. You will probably have met this form of number representation in maths already: 12,000,000 is written 1.2×10^7 and 0.00012 is written 1.2×10^{-4}. In the above example 1.2 is called the mantissa, and -4 is the exponent. Mantissa and exponent are usually packed together in one store, or one pair of stores, the unpacking being carried out automatically by means of appropriate routes in the dataflow. A typical floating-point format for a 32-bit store allows 8 bits for the exponent and 24 bits for the mantissa giving an accuracy of about one part in 8,388,608 (the most significant bit in the mantissa is reserved for the sign). When floating point numbers are used in arithmetic then any surplus digits are discarded off the least significant end. Example: suppose that we only have 3 digits in which to hold the mantissa, then 1.23 x 4.56 = 5.60 which is only roughly correct. This answer is in error by 0.0088, which is an error of approx 0.2%. In practice we have around 9 or 10 decimal digits available, but nevertheless the same applies — we have a slight error. Languages such as FORTRAN have a facility for doubling the number of digits available for arithmetic variables and working, however this halves the capacity of the computer because two stores are used in place of every one.

Let's look now at the sources of error in computing. They arise from (a) mistakes, (b) rounding and truncation (c) approximations of the method of solution. I will examine each source of error. MISTAKES need no further mention except to outline the places that most mistakes occur:

i) Transportation of digits. Example writing 6437 instead of 6347.
ii) Repeating the wrong set of digits. Example writing 46621 instead of 46221.
iii) Mistakes in the location of the decimal point and also omission of the negative sign.
iv) Remembering constants incorrectly. Example writing pi as 3.14259 instead of 3.14159.

ROUNDING AND TRUNCATION ERRORS:
Two terms are needed; they are:-
Absolute error which is real value — rounded value
Relative error which is absolute error/real value.
To see how relative error gives a better indication of the significance of an error, consider the following examples: If distance is measured as 3000 km with an error of ±5 km, then the absolute error is 5km which is quite large, but the relative error is 5/3000 = 1/600 which shows that the measurement is quite precise. On the other hand suppose that a quantity is measured as 0.00007 m and the error is ± 0.00001m, then the absolute error is 0.00001m which is quite small, but the relative error is 0.00001/0.00007 = 1/7 which is rather poor.

Errors accumulate when an arithmetic operation is performed. The way they accumulate is shown below. Error in addition/subtraction is the sum of the absolute errors. Whereas error in multiplication/division is the sum of the relative errors. An example is given below.

If a farmer measures his rectangular field as 100m by 50m correct to the nearest metre, then the absolute error is 0.5m in each measurement. If we wish to calculate the error in measuring the perimeter of the field we must add the absolute errors i.e. 0.5 0.5 0.5 0.5: 2.0m This means that if the farmer wishes to buy fencing to surround his field then he must buy 300m; an extra 2m to allow for the error in measurement, that is he must buy 302m of fencing. To calculate the area of the field the dimensions must be multiplied, and so we must add the *relative* errors, the relative errors are 0.5/100 = 0.005 and 0.5/50 = 0.01 and so the relative error in area will be 0.005 + 0.01 = 0.015. If we know the relative error in 100 x 50 = 5000 sq m is 0.015 then the absolute error can be found as 5000 x 0.015 = 75 sq m; and so the farmer could be in error by as much as 75 sq m in area; the area of his field could be anywhere between 4925 and 5075 sq m and all because he measured his field to the nearest metre.

To summarise: when performing addition or subtraction add the *absolute* errors. When performing multiplication or division add the *relative* errors. Errors can also accumulate in powers, roots and other functions; this is left to the reader to find in the references if he is interested.

The point of this article is to make the computer user aware of errors. Be careful when you get an answer from a program, don't assume that because it is printed to nine figures it is correct to all those figures. You should be able to calculate (in the program) the accuracy of your quoted results, and print them thus 5000±75 sq m. Another example may be where you measure an angle using an ordinary protractor as 30°, when it is difficult to measure to an accuracy of more than ½°, and yet quote the cosine of that angle as 0.8660. You should quote it as 0.866 ±0.004, the fourth decimal is meaningless.

An understanding of accuracy may help to have avoided the well known case of the person who received a bill for £0.00. A few weeks later there came a final demand for £0.00. Apparently he wrote a cheque for that amount and posted it; no more demands were received.

REFERENCES (for further reading)

Conte S D	Elementary Numerical Analysis	McGraw-Hill 1965
Fox L & Mayers D F	Computing Methods for Scientists and Engineers	Clarendon Press 1968
McCracken & Dorn	Numerical Methods and FORTRAN Programming	John Wiley 1964
Butler & Kerr	An introduction to Numerical Methods	Pitman 1962

COMPETITION

Write a program in BASIC to read an integer between 0 and 3000 and convert it to Roman numerals.

Example: DATA 1978
OUTPUT: MCMLXXVIII

Two prizes will be given:

1) For the shortest fully working program : PRIZE £5
2) For the best documented and most easily understood program. (Program must include validation checks) : PRIZE £10.

Send SAE if you wish your programs to be returned.
Address: Sheridan Williams, 114 Beech Road, St. Albans, Herts. AL3 5AU.

David McBride (age 14)

APPLICATIONS OF COMPUTERS

The computer is a familiar tool to some people, yet others know hardly anything of its applications, many of which provide efficiency and pleasure, as well as speedy service to very many people. The following are some aspects of the uses to which we put computers.

Computers in banks, save the clerks, hours of labourious work, totaling accounts, and checking balances. The computer, using a Magnetic Ink Character Reader, can debit a cheque from an account automatically, and at great speed. This is extremely useful in a country where 15,000,000 cheques are made out daily. In addition to the basic numerical aspects, the computer is often programmed to produce the various letters of business for the bank, i.e. statements, reminders etc.

Lately in the press have come reports of the long arm of the law extending into computing. Not only can files on the criminal fringe be compared from Police stations many miles apart, but so can finger-prints, the computer converting each print into a digital code, then comparing codes. The final decisions as to prosecution, and of course the 'leg work' must still be done by the Police.

For quite a few years now computers have been becoming more and more involved with selling and ordering. In some of the larger supermarkets, all ordering can be done via the computer terminal. Orders may be dispatched, the goods often turning up in a couple of days, which is very important in these competitive days. Invoices often come from the hard copy terminal, which means that the chances of human error are reduced.

In the USA a number of experiments have been carried out into the use of computers as an aid to learning. The terminal becomes a 'teaching machine' and presents lessons to children who respond to questions the machine asks, then adjusts its teaching rate accordingly. Automatic marking is a strong incentive to the pupils to work hard, as is the option to use a computer. More recently these experiments have led to the introduction of small dedicated calculators, which produce random number questions to demand, two common examples are the Texas Instruments 'Little - Professor' and Arnold's 'Digitor'.

Various computer applications are found in libraries. Most large libraries have their catalogues on micro-film, read by complex viewing systems. This initial file is computer produced. A library with a computer on site, which has time to spare, could do away with tickets totally. Merely by running a light-pen over a label in a book, that copy could be marked 'loaned' or 'returned' inside the computer.

A number of Automatic devices are now in fact controlled by a computer. Examples are Railway Signalling systems, Traffic lights, Auto-pilots, even the common doorbell is becoming automated. These devices are dedicated computers in the main, where programming is accomplished by way of hard wiring or by the use of micro-circuits containing a program (R.O.M.).

It is becoming common now for computers to have some form of graphic display, either the V.D.U., on personal computers, or a graph plotter on commercial models. These are both capable of forms of drawing, and cartooning, but the incentive is of course human. The same applies to Computer poetry, which tends to follow a set of rules, to allow humans to understand it. One remembers the computer programmed to translate 'Out of sight, out of mind' into Russian. It produced 'Invisible, insane', which is a perfect translation, but misses the point of the saying. This emphasises the incapacity of the computer to deal with emotional quantities. A similar effect is found in Computer music, which is either totally random, or bound tightly to the rules of the program.

Computing is now turning up in hospitals, in both a diagnostic role, and an information storage role. Records of treatments, drugs, vacant beds, pending patients, etc. all fall in the area of the hospital computer.

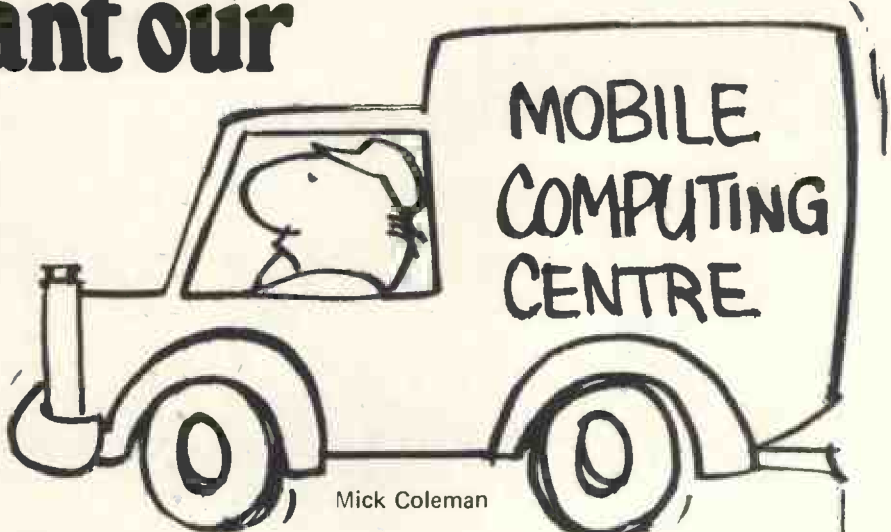
Real time computing is coming to the fore with the introduction of Microprocessors. A number of instruments make full use of the processing capability of these devices. In situations where fast output to certain data is required real time computing is the only solution. Examples would include Airline Bookings, point of sale terminals, intensive care units, pressure gauges, and a number of other things.

Small units with a magnetic tape or paper tape program can control various industrial machines. Drills, or lathes, punches, or milling machines can be adapted to work from M.P.U. based instructions. These instructions include depth of cut and position of cut.

For those of us who regard these applications as examples of the displacement of people by computer, then we should remember that before the advent of the mechanical digger people were forced by want of food, to dig manually through mountains. No one today wishes to see those days back, nor should we regret the introduction of the computer.

David McBride (Bottom Row, Centre) is a pupil at St. Patrick's R.C. Comprehensive School, Thornaby on Tees, Co. Cleveland. The two boys seen in the photo at the back row, left, are Anthony Walsh and Thomas Bennington. They completed successfully in the Kent Software Trophies competition organised by the University of Canterbury, winning a fourth prize with their flowchart to aid passenger enquiries for British Rail. They are both only fifteen and the competition was for students aged eighteen or over. Their teacher, Mr. J.H. Telford, encouraged David to submit his entry to PCW.

Do we want our schools to have A PERSONAL PROBLEM?



Mick Coleman

"We stand at a **turning point** in the history of computing in Secondary education. We have passed through the eras of visiting Data Processing departments to gape at tape drives clicking round, and of carrying suitcases of marked cards to a friendly company five miles down the road, and we are now coming to the end, I think, of 'cheap' computing by telephone. What comes next is surely the personal computer in school and college . .".

So wrote Charles Sweeten, secretary of MUSE, in his article 'Do we want our schools to be personal or terminal' which appeared in the inaugural edition of PCW. The quotation has been carefully chosen since it highlights, in a single paragraph, the two great dangers now present in the area of schools computing.

The first danger is the LEAs, (Local Education Authority) and possibly even the schools, will fail to recognise that the turning point has arrived. And yet the facts are indisputable. The computer, over the past two decades, has become virtually omnipresent, influencing so many facets of our daily life. Over the next 20 years this process will accelerate continuously so that, by the year 2000, we will be as dependent on the computer as we are, say, on oil today. The majority of our school-leavers will obtain jobs which involve their working with computers in some capacity — simply because few jobs will exist that do not have such an involvement.

It will, therefore, be as necessary to teach children the fundamentals of computing as it is to teach them the fundamentals of subjects such as Mathematics and English today. This has a considerable implication in terms of numbers. At present, it is common to think in numbers of 20 or 30 students. When those numbers are multiplied by 10 — as they will be — and apply to *every* Secondary school, then the problem of providing computing facilities becomes enormous.

Herein lies the second danger: that since the advances in micro-processor technology have made it possible for every school to have its own computer, then the decision will be made to give every school its own computer. This approach, both financially and educationally, is a potential disaster.

Consider the financial implications. In many ways, the personal computer boom is a demonstration of history repeating itself. Micro-systems are being sold, complete with varying amounts of software, in much the same manner as were mainframe systems 10 to 15 years ago. As then, such software as is available is relatively primitive (an educational problem!). But this situation will change. As demands increase, the manufacturers of micro-systems will produce wider ranges of bigger and better input/output routines, applications packages, compilers.

They will then discover that producing software has become more expensive than producing hardware and 'unbundle'; that is, sell hardware and software separately. Worse, they might well adopt the policy of the major computer manufacturers and license their software annually.

A global policy which puts a personal computer system into every school and college will require not only a commitment by LEAs to purchase the initial system but also an on-going commitment to provide cash for software *for every site.* The possibility that a machine might sit idle, through lack of software, is only too real.

Such a situation would, of course, be educationally undesirable; but this is a problem for the future. Of more immediate concern is the less obvious difficulty of the 'lone-terminal' syndrome.

As Charles Sweeten pointed out in his article, a common approach towards the provision of schools computing facilities has been the installation of a single terminal in a school, the terminal being connected by telephone line to a distant mainframe. In theory, this sort of facility can be used to support a course in Computer Science for a reasonable number of students. However, as with any subject, a class will be made up of the enthusiastic and the not-so-enthusiastic. What happens with the lone-terminal in practice is that it is used predominantly by the enthusiasts. The less-enthusiastic are unable to get near the terminal, they fall behind, become even less enthusiastic and a vicious circle has been created. In the end, the provision of the facility becomes counter productive, the full manner of its use having been discovered by the few, the majority viewing the terminal as an object of mystery. The fear is that replacing the lone-terminal by a £2000 configuration of micro-computer, control terminal and disc will merely exacerbate matters, fostering an identical situation with more complex equipment. Since the prime objective in teaching Computer Science is to dispel the air of mystique surrounding computers in general, this effect is terrible.

If Computer Science is to be adequately taught to a class, then facilities *for a class* must be available. It is inconceivable, for instance, that a woodwork class be equipped with only one saw, one chisel etc,; it should be equally inconceivable that computing facilities be provided that are unsuitable for class use. For an in-house configuration, therefore, we are talking about either a multi-terminal system, or a system that can support a significant capability for the running of batch jobs. Ideally, we are talking about a system that can provide both. Thus, what is educationally desirable is likely to be financially impracticable — excluding the earlier point about future trends in the pricing of software. Even if the money is made available, it must be questioned whether every school has the desire to run its own computing service, coping with the attendant problems of maintenance, documentation, advice and so on. However, having argued that personal computers in every school and college are not the answer, I am sure that the personal computer must form an essential part of the solution to the schools computing question.

One clear indication, for instance, of the microprocessor revolution is that it now enables relatively inexpensive **Computing Centres** to be created *specifically for schools' use.* This certainly seems to be a realistic way in which a classroom's worth of terminals could be made available to every school. Such centres could be based at selected schools or, as the result of some collaborative arrangement, at local computer installations. Polytechnics or Universities for instance would be ideally suited for such an arrangement. Wherever it was located, the centre would be responsible for serving the requirements of a number of schools both in terms of facilities and advice.

The use of the term 'centre' has, in the past, implied that the computer user visits the computer facility, the latter being immovable. Again, the advent of the microsystem has been particularly notable for the fact that they can be transported from place to place without problem. Thus, we could well see the large-scale emergence of **Mobile Computer Units**. Vans, akin to those employed by mobile libraries, could be equipped with anything from a simple system to one comprising 20 or more computer terminals. Used in combination with a Computing Centre, this approach is highly attractive. Given that it is now possible to buy limited facility microprocessor based terminals — such as the Commodore Pet — for less than £750, mobile units could be a highly cost-effective way of providing class teaching systems for schools use.

There is no doubt that the micro-computer is the means by which effective shcools computing facilities can be provided. To quote Charles Sweeten again, "We stand at a turning point in history of computing in Secondary Education". Given that the right road is taken, schools computing need never look back.

THREE NOVELTY PROGRAMS FOR A POCKET CALCULATOR

Derek J. Chown

I am constantly amazed at what power is available in pocket calculators for only a few tens of pounds. While programmable calculators are definitely not computers they do facilitate real programming with conditional branching, loops, and subroutines. The author has tackled the following number problems on his T157.

i) Find the prime factors of any whole number.
ii) Find the sequence of prime numbers starting at any predetermined point.
iii) Find the right-angled triangles whose sides are whole numbers.

i) Prime Factors

The method is to try 2 as a factor. If it is, then divide the number by 2 and "pause" with 2 in the display. The program then tries 2 again. If 2 is not a factor then the program proceeds to try all odd numbers as it is not practical to try only the prime numbers. If the number itself is found to be prime then the display flashes.

STO 1	Store the number whose factors
STO 4	are to be found.
CLR	
STO 0	Clear store 0 ready to receive the sum
STO 7	of the factors. This provides a check.
1	
STO 2	1 in register 2, for incrementing register 3.
2	
STO 3	2 in register 3. Trial factor.
2nd Lbl 1	
RCL 4	
÷	
RCL 3	
–	
STO 5	Store result of division just in case register 3
RCL 3	contains a factor.
=	
2nd INV $x \geqslant t$	If display <0 then there are no more
GTO 3	factors, so go to label 3.
2nd INV Int	Fraction part.
2nd $x = t$	If fractional part is zero, register 3 contains
GTO 0	a factor, so go to label 0.
RCL 2	
SUM 3	Increment the trial factor.
2	The increment for the first loop only is 1.
STO 2	Thereafter it is 2.
GTO 1	Go back to try next odd number as a factor.
2nd Lbl 0	A factor is in register 3.
RCL 5	Prepare to work on the new number
STO 4	produced by dividing by the prime factor.
RCL 3	
2nd pause	Display the factor.
SUM 0	Add the factor to the number in register 0.
GTO 1	Try again with same trial factor.
2nd Lbl 3	No more factors.
RCL 1	
STO 7	Prepare to compare original number with
RCL 4	number in register 4.
2nd $x=t$	If there are the same no factors were
GTO 8	found so go to Label 8 to flash display.
SUM 0	Otherwise, displayed number is also a
R/S	factor so add it into register 0, and stop.
RST	Reset ready for next run.
2nd Lbl 8	The original number was prime.
GTO 9	So create error condition by branch to
RST	non-existent label to make display flash.

ii) The Sequence of Primes

To find the first two primes greater than a million. Simply key in 1000000 Press RST, and start the program. Five minutes later we have 1000003, and another eight minutes produces 1000033.

The program first makes the keyed-in number odd and then tries all odd numbers up to the square root as factors. As soon as a factor is found the number is discarded, but if a prime is found the program stops.

STO 6	Temporary store for keyed number.
CLR	Clear machine's current calculations.
STO 7	Clear test register.
RCL 6	Retrieve keyed number.
÷	
2	
=	
2nd Int	Take integer part.
X	
2	
+	
1	
=	Ensures an odd number.
2nd Lbl 7	
STO 4	Store trial prime.
3	First trial factor
STO 3	in register 3.
2nd Lbl 1	
RCL 4	Trial prime
÷	
RCL 3	Trial factor
–	
STO 5	Store quotient
RCL 3	
=	
2nd INV $x \geqslant t$	If quotient $<$ trial factor then square
GTO 3	root has been passed, so go to label 3.
2nd INV Int	Fractional part.
2nd $x = t$	If fractional part is zero then a factor
GTO 0	has been found, so reject number.
2	Increment the trial factor by 2
SUM 3	by adding 2 into register 3.
GTO 1	Go to label 1 to try next factor.
2nd Label 3	
RCL 4	
–	
1	
=	This is a "trick" because so far the
2nd $x = t$	program would pick 1 as a prime
1	and miss 2, so if the number is
+	1 add 1 to it and get 2.
1	
=	
R/S	Stop to display prime.
2nd Lbl 0	
2	Increment the trial prime by 2.
SUM 4	
RCL 4	New trial prime.
GTO 7	Go back to test new trial prime.

iii) Most readers will know that a triangle with sides in the ratio 3:4:5 has a right angle, some will also know that a 5:12:13 triangle also has a right angle. There are many others like this, possibly infinitely many though I haven't managed to prove this. I have managed to prove, however, that the longest side is always odd except for cases like 6:8:10 which is just a magnified 3:4:5 triangle. This proof enabled me to write a faster program, but omitting triangles where all the sides are even, like 6:8:10.

The program starts with a hypotenuse (longest side) of specified length. It must be odd and greater than 2. The program pauses to display the sides, shortest first, longest last, and then stops with the smallest angle of the triangle displayed. This angle enables the user to see quickly whether the triangle is a magnification of a smaller right angled triangle. The sides can be reviewed by keying GTO 3, R/S. To continue to the next triangle press R/S.

STO 3	Store minimum hypotenuse.
CLR	Clear uncompleted calculations, etc.
STO 7	Clear test register.
2nd Lbl 1	
CLR	
STO 1	Zero to register 1.
RCL 3	
x^2	"The square on the hypotenuse"
STO 6	
÷	
2	
=	This is the largest possible square on the
$\sqrt{x}$	shortest side, so $\sqrt{x}$ gives the longest value
2nd Int	of shortest side, but must be a whole number.
STO 0	Store max length of shortest side.
2nd Lbl 0	Register 0 is used to count the number of loops.
RCL 6	Hypotenuse square. (Call this h^2).
–	
1	
SUM 1	Increment shortest side.
RCL 1	Shortest side is 1 first time round.
x^2	shortest side squared (call this s^2).
=	This gives $h^2 - s^2$ = middle side squared. Call this m^2
$\sqrt{x}$	Length, m, of middle-length side.
–	
2nd Int	
STO 2	Store integral part of m.
=	Fractional part of m.
2nd $x = t$	If fractional part is zero . . . .
SBR 3	call subroutine 3.
2nd DSZ	Decrease contents of register 0 by 1.
GTO 0	If result is > 0 go to label 0
2	Increment h. Could be 1 if all
SUM 3	triangles like 6:8:10, 12:16:20
GTO 1	are required.
	Go back to try new h.
2nd Lbl 3	Subroutine 3. Triangle found.
RCL 1	s, shortest side.
2nd pause	Display shortest side.
÷	(for working out tan of smallest angle)
RCL 2	m, middle-size side.
2nd pause	Display m.
=	Give tan of smallest angle.
STO 5	Temporary storage.
RCL 3	h, the hypotenuse.
2nd pause	Display h.
RCL 5	
2nd INV tan	Inverse tangent give smallest angle.
STO 5	For reference.
R/S	Stop.
INV SBR	End of subroutine, return.

GOOD VALUE FOR MONEY

Mike Dennis

TANDY TRS-80 Review

PCW Our special thanks to Barney Price of Tandy for the whole-hearted cooperation given us. Readers will note that this evaluation is not entirely uncritical, and we congratulate Tandy on its enlightened attitude. **PCW** welcomes the TRS-80 to the ranks of value-for-money machines.

Introduction

The Tandy TRS-80 is a microcomputer system that has clearly been designed to blend in with the decor, to provide the user with practical computing ability and with the minimum of effort. It is a computer for the person who doesn't want to be bothered with busses and buffers and just wants to get on and write programs. I have tried to evaluate it with this in mind.

The Tandy microcomputer system arrives carefully packed in three boxes and is a 'smart' system. What this means is that there is an interpreter program already installed in ROM. As soon as you switch on, you can immediately start writing programs in a high-level language. Usually, this language is BASIC and the Tandy is no exception. The model that I had for review was one from the budget end of the range with Version 1 Basic and 4K of RAM. Unfortunately, I couldn't have the machine for very long and as a result this review is not as exhaustive as I would have wished.

Upon unpacking, one could be forgiven for wondering whether or not there was a bit missing because Tandy have very neatly packaged all the main 'guts' underneath the sloping keyboard. This has resulted in a very compact assembly in a case belonging to the hydrocarbon world. The styling is vaguely futuristic and has obviously been designed to be aesthetically pleasing in the home. It will be a question of personal taste whether you or your wife like it. Where I personally feel the entire concept has come unstuck is in the inevitable birds nest of cables as a result of connecting up these units. There are the three mains leads (which come with American style plugs) plus the various leads carrying the logic and power supplies. Also, this multi-unit approach does mean that transporting the computer from room to room becomes a bit of a chore as you need a lot of hands! I prefer the more 'integrated' design style for the front parlour.

Connecting Up

Connecting up is both 'fool' and 'idiot' proof. The computer connects up to the power supply, video display and cassette recorder via three 5-pin DIN plugs. As you can plug any lead into any socket, this could be a potentially disastrous situation. Tandy, however, have chosen different combinations of pin connections for each socket to prevent any mishaps and it works — I tried it!

Power

The power supply is nothing more than a transformer and some rectification and supplies 16v AC at 1Amp and 18v DC at 350mA to the computer alone as the video display has its' own internal supply. All the actual smoothing and regulation is done on-board in the computer itself. I think this approach *good* in their particular design context as it removes a bulky object from the case and thus allows the packaging to be that much smaller. The transformer is double-wound for electrical safety and so the mains lead is two core.

Cassette

The cassette recorder is just an ordinary general purpose 'cooking' version and comes under the 'Realistic' label. I expect that you may have one already. It offers a remote stop-start facility and this is used by the computer during tape handling. You have to remember to:-

a) Stop the machine after loading or dumping to prevent 'flats' wearing on the pinch-roller.

b) Remove the Remote jack from the recorder if you want to rewind or fast wind the tape. This is only a minor irritation and virtually all remote operated recorders do this and it is a feature of the recorder and **not** the computer. If it was my system I would probably put an over-ride switch in the recorder.

Connection is made via three miniature jack sockets and a special lead (supplied) that converts to the DIN socket on the computer. Table 1 shows the pin connections. Data transfer rate is 300 baud but not Kansas City CUTS.

Video Display

The video display is housed in a normal television receiver case. The video lead comes out from the front of the case where the volume control would normally have gone. 'V' obviously either means Volume or Video! I would have liked this lead to be at least double the length provided which was only about 18". If you are going to adopt the unit approach then it seems reasonable to expect a long lead to give you the chance of arranging the system more conveniently.

The video signal itself couples into the display via an opto-isolator. The video information is almost the professional broadcast standard of 1v peak to peak when terminated by 75 ohms and with negative going syncs. I tried feeding this signal into various television monitors with varying degrees of success. This was due to the unfortunate fact that the Tandy appears to generate a video signal with a field rate of 60Hz and not the standard in this country which is 50Hz. The effect on most monitors was a loss of vertical hold. Those monitors which were specifically designed to accept either standard locked up satisfactorily. This would also explain why some people experience a 'hum-bar' or dark bar which slowly rises up the screen. I received the circuit for the TRS-80 after the review sample was returned and so was unable to try the effect of one of the preset controls in the computer itself to see whether or not it was possible to adjust the field rate to 50Hz. Obviously, since the system is sold with its own display, you may well argue that my point is rather academic. However, I suspect that there may be a potential market who may well have wanted the facility of using the TRS-80 as a caption generator in CCTV applications or wanted to feed the output to a number of monitors as a visual aid when teaching BASIC programming. They may find that it won't work. Inside, there is very little else apart from the power supply and some scanning circuitry. The internal construction was that of 'yer average tele'.

To be frank, I found the performance of the display disappointing. The screen could have been a bit more sharply focussed and there was some ghosting on the characters. As there was no clamp or dc restoration of the signal, the setting of the Contrast and Brightness controls were fairly critical. Altering the brightness control to make the background grey (as suggested in the manual) resulted in an objectionable flutter on the display becoming visible. The characters themselves also tended to jitter up and down albeit by a small amount. I have seen a stable display on a different sample so perhaps it's only a question of checking and tweaking the displays before they leave the show. However, I do think that Tandy ought to look at this problem. The character format is 64 x 16. I have read that it is software controllable to 32 x 16 but I didn't come across this in the manual. There is a quite versatile graphics capability provided which gives a matrix of 128 x 48.

One feature I liked was the provision of an edge connector for further expansion. Most of the CPU signals are present together with RAS, CAS etc for dynamic RAM. All the signals are fully buffered. Five volts is also provided but at 'limited current' which alas is unspecified. $\overline{\text{IORQ}}$, $\overline{\text{MREQ}}$, $\overline{\text{RD}}$ and $\overline{\text{WR}}$ have also been gated together to provide $\overline{\text{IORD}}$, $\overline{\text{IOWR}}$, $\overline{\text{MRD}}$, MWR. I can't really see the point of this as it doesn't save any pins on the connector. I usually prefer to keep my options open and keep the signals separate. There would appear to be no system clock available on this connector which would be annoying should you need it!

The keyboard was OK apart from the 'graunchy' noises some of the return springs made as the keys were released though this could have been specific to my sample. Certainly, the keyboard has a much better feel to it than has the PET. I occasionally got some double-entries.

Software

As mentioned earlier, the TRS-80 is 'smart' and expects you to talk BASIC to it. You cannot enter any program in machine code or any other language and many of you will be happy about that! However, there is a monitor ROM (available sometime) called T-BUG which allows you to write in machine-code. Machine-code is particularly useful for 'talking' to your own peripherals - BASIC usually is not.

Tandy have tried to make the whole operation as simple as possible. One feature that I liked very much was the ease of deleting wrong keystrokes. There is a key labelled ← which does precisely that: back-steps and erases the last character. No more having to remember whether it should be Control D or whatever. Nice one, Tandy.

Version 1 BASIC is fairly elementary. The available statements are similar to Tiny Basic but with the addition of floating point, two string variables and graphics statements such as Set, Reset and Point (test) a specific location on the display. One minor trauma with the graphics is that location O,O is at the top LH corner of the screen and not the bottom LH corner as you would expect on a graph. This is a very minor point as the position of the origin can be easily modified by the program.

You can use the computer in 'calculator' mode by entering PRINT followed by the equation. By some quirk, there are some minor errors involving numbers around the figure 10. I give a few examples below:

10.000005 + 3 x 3 = 10
10 + 3 x 3 = 10.000008
10.000001 + 3 x 3 = 10.000008
9.999999 + 3 x 3 = 10.000005

Obviously, this must be due to rounding up by the algorithms but if you repeat the sums with a pocket calculator and round up the intermediate results, the final result is not the same as the TRS-80. All other calculations that I tried were very accurate and rounded up as expected. Most strange.

As there are no trig., log., squares, powers or square root statements in this version, sub-routines are provided in the back of the handbook. They proved to be pretty accurate and included the correct signs in "All Stations To Crewe" as my old maths master used to say! Unfortunately, loading in the TAN sub-routine also necessitates those for COS and SIN and so you eat quite considerably into your available RAM. But twenty-seven of the normal statements can be abbreviated thus retrieving some of this loss. Some further savings can also be made with the TRS-80 BASIC as it allows you to put more than one statement on a line. However, the available RAM should be *more than ample* for your early programs. This "loss" of available memory is really a fact of life when one uses fairly elementary BASIC interpreters and in no way detracts from the TRS-80. Any answers out of the normal integer range of 32768 are converted automatically into scientific notation before display. The range of scientific notation is 1E ± 38.

In early versions of Palo-Alto Tiny Basic, there was one small bug which was corrected in later versions. The programme line that reveals this bug is "For A = 1 to 32767" which never ends. As a matter of interest, I tried running a program with a similar instruction and although only of little consequence, it didn't finish either. If you have a TRS-80, try this small program:-

```
10 FOR A = 1 TO 32500 STEP10000
20 PRINT A
30 NEXT A
40 END
```

You should get printed 1, 10001, 20001 and 30001 in theory.

Program debugging itself is facilitated by being able to examine the variables with a PRINT statement after a program break. I would have liked to have seen the ERROR statements summarised in the otherwise *excellent* program statement list inside the back cover of the handbook. You can list the program both from the beginning and from a specified line. As some programs may have more lines than there are lines on the display, it is sensible to inhibit scrolling and this is what the TRS-80 does. Hidden lines can then be displayed sequentially by using the ↑ key. When you reach the statement that you wish to alter, you must remember to hit the ENTER key before writing the new line otherwise it won't go in. I would have preferred some form of visual prompt to remind me.

I ran the bench mark programs given in PCW Vol 1 No 1 and have tabulated the results in Table II. Space does not allow me to reprint the actual programs again. The results were a trifle slow when you compare them with other machines running a similar BASIC. The most likely reason for this must surely be the slow clock frequency which measured about 1½MHz.

I tried to corrupt the program with mains interference by switching motors, large transformers, and other inductive loads on and off but with no effect. One quirk which happened and which I was unable to reproduce was a sudden doubling in height of the display characters with PIDS in the middle of the screen! Needless to say, the computer was totally inoperative in this condition and I had to switch it off to clear the fault being none the wiser as to the actual cause. Normal resetting could be a problem with dynamic RAM if you don't design it carefully. Tandy *have* done so and use the NMI input (and some software presumably) driven by a push-button at the rear of the machine. It does mean that you can't use the NMI yourself and to this end, it is one of the Z-80 signals that is not brought out to the rear expansion socket.

The documentation was generally excellent being in the form of a *very good* introduction to BASIC although I expect that some may find the American approach rather over-powering to say the least. I did find it extremely tedious searching through the book for individual chapters. It would have been far better if Tandy printed the chapter number in front of each page number. Just a number would have sufficed, e.g. 20 - 130 for Chapter 20 - Page 130, to make life a bit easier. I wish manufacturers would also include circuit diagrams as part of the documentation. I am afraid that Tandy do not provide a circuit or a software listing of their BASIC ROM.

Summary

To be fair, one should examine this machine within its design context and as a machine to get you computing with the minimum of effort, *it most certainly succeeds.* Tandy have gone to a lot of trouble to make computing as easy as possible. I have some reservations regarding the package as a whole as you almost certainly will have your own cassette recorder. If a small UHF modulator was included in the case and some slight adjustments made to the display field rate then you could feed the TRS-80 into your own domestic television. If the computer was then sold as a stand-alone device, the extra money could either go towards extra facilities, such as Version 2 BASIC, or a price reduction. The computer would then offer exceptional value for money.

Nevertheless, at this moment in time, the Tandy TRS-80 is still probably the cheapest way to wake up 'smart' and thus provides good value for money.

TABLE I

Pin No.	Function
1	Remote
2	Ground
3	Remote
4	Cassette Output
5	Cassette Input

TABLE II

Bench Mark Program	Time
BM1	2.5 secs
BM2	18.0 secs
BM3	34.5 secs
BM4	39.0 secs
BM5	45.0 secs
BM6) without line	67.0 secs
BM7) 430	109.0 secs

Appendices

1/ If the pinch-wheel of any tape recorder is kept pressed against a stationary capstan, there is the risk that a flat will be imprinted on the pinch-wheel. This will cause perturbations (good word!) to the speed of the tape past the heads. The audible effect of this is wow or flutter. As far as digital recording is concerned, it will depend on the type of data acquisition circuitry used and error checking as to whether this will have any effect or not.

2/ Most monochrome television receivers and some colour receivers for reasons of economy (cheapness) do not have mains transformers. It is therefore possible for the chassis of the set to be at mains potential. This is generally safe enough until you want to connect up to your hi-fi or computer. The Tandy display is no exception. One way around this problem is to fit an opto-isolator which effectively couples the video signal to the set but isolates the computer from the set chassis. Aerial sockets perform the same function being rather special beasts but of course you need a modulated version signal to feed in. *On no account* should you try and modify your own domestic television if you are in any doubt as to your competence and/or skill at artificial resuscitation and heart massage.

PCW Reader Freddie Nicholls of the Optronix Co., 1 Strawberry Vale, Twickenham, has had his TRS-80 updated to Level 2 BASIC. He likes it, saying that the only criticism he has is that its impossible to define a function. Approximate cost of up-date was £70. He also has T-BUG (new) enabling amongst other things programming in Hex for the Z-80 mpu. T-BUG runs on both Level 1 & Level 2 systems. Barry Nicholls says he's very pleased with his TRS-80 running Level 2 BASIC. PCW

MK14–the only low-cost keyboard-addressable microprocessor!

The new Science of Cambridge MK14 Standard Microprocessor kit

Just £39.95 (+ £3.20 VAT, and p&p)

The MK14 National Semiconductor Scamp-based Microprocessor Kit gives you the power and performance of a professional keyboard-addressable unit – for less than half the normal price! For less than £44.00 you can have your own microprocessor. One with a specification that makes it perfect for the engineer who needs to keep up to date with digital systems, or for use in school science departments. It's ideal for hobbyists and amateur electronics enthusiasts, too.

But the MK14 isn't just a training aid. It's been designed for practical performance, so you can use it as a working component of, even the heart of, larger electronic systems and equipment.

MK14 Specification

* Hexadecimal keyboard
* 8-digit LED display
* 512 x 8 Prom, containing monitor program and interface instructions
* 256 bytes of RAM
* 4MHz crystal
* 5V Stabiliser
* Single 6V power supply
* Space available for extra RAM and RAM I/O

Free Manual

Every MK14 Microprocessor kit includes a free Operation Manual. It contains operational instructions and examples for training applications, and numerous programs including math routines, timing, general purpose sequencing, games, etc.

Designed for fast, easy assembly

Each 31-piece kit includes everything you need to make a full-scale working micro-processor, from 14 chips, a 4-part keyboard, display interface components, to PCB, switch and fixings.

The MK14 can be assembled by anyone with a fine-tip soldering iron and a few hours' spare time, using the step-by-step illustrated instructions provided.

Tomorrow's technology – today!

"It is not unreasonable to assume that within the next five years...there will be hardly any companies engaged in electronics that are not using microprocessors in one area or another."

Phil Pittman, Wireless World, Nov. 1977

The low-cost computing power of the microprocessor is already being used to replace other forms of digital, analogue, electro-mechanical, even purely mechanical forms of control systems.

The Science of Cambridge MK14 Standard Microprocessor Kit allows you to learn more about this exciting and rapidly advancing area of technology. It allows you to use your own microprocessor in practical applications of your own design. And it allows you to do it at a fraction of the price you'd have to pay elsewhere.

Getting your MK14 Kit is easy. Just fill in the coupon below, and post it to us today, with a cheque or PO made payable to Science of Cambridge. And, of course, it comes to you with a comprehensive guarantee. If for any reason, you're not completely satisfied with your MK14, return it to us within 14 days for a full cash refund.

Science of Cambridge Ltd.,
6 Kings Parade,
Cambridge,
Cambs., CB2 1SN.
Telephone: Cambridge (0223) 311488

To: Science of Cambridge Ltd, 6 Kings Parade, Cambridge, Cambs., CB2 1SN.

Please send me an MK14 Standard Microprocessor Kit. I enclose cheque/money order/PO for £43.55 (£39.95 + 8% VAT and 40p p&p).

Name ______________________

Address *(please print)* ______________________

______________________ Allow 21 days for delivery.

Science of Cambridge

HCW

A FIRST COURSE

Bits and Bytes

Robert Henley

In everyday life and indeed even in the realms of advanced mathematics we count using a system which has come down to us from antiquity. We use the decimal system in which ten symbols are used to represent any quantity, great or small between —oo and +oo. This system strikes a reasonable balance between economy of symbols and a compact representation of large quantities, as we shall soon appreciate when we consider other number systems. The only disadvantage of the decimal system as far as computing is concerned is that it poses a formidable problem for the hard-ware designer.

This difficulty derives from the fact that it would be necessary to design a system which could electrically represent the ten quantities 0 — 9. In electrical circuits the two circuit states 'on' and 'off' are considerably more reliable. A transistor circuit can be designed with comparative ease to have *two discrete states*; and similarly where signal voltages or currents are concerned, the two states of 'signal' or 'no signal' can be achieved with little ambiguity even in the most hostile electrical environments.

These considerations have led — from the earliest days of computer design — to the use of the binary number system . Although the binary number does not embody the economy of the decimal system particularly in the expression of large quantities — in fact it is positively clumsy as far as the human user is concerned — it does over-come the problems of stability which attend the use of the decimal system in electrical circuits.

Counting in the binary system proceeds in exactly the same way as in the decimal system except that we 'carry' to the next column when the count reaches 2 instead of 10. Only two symbols are required 0, and 1.

In the decimal system each column represents a power of 10. Thus, if the digit 3 appears in the *third* column to the left of the decimal point then it represents 3×10^2. The same digit 3 in the *first* column to the left of the decimal point represents $3 \times 10^0 = 3$. Similarly, the columns to the right of the decimal point represent *negative* powers of 10. The first column represents 10^{-1} and so on; thus a 3 in this column represents $3 \times 10^{-1} = 0.3$.

In *binary* notation the corresponding columns represent powers of the base 2. For example the binary number 1101.11001 represents:—

$1 \times 2^3 + 1 \times 2^2 + 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-5}$ which is the binary equivalent of 12 25/32 = 12.78125 in the decimal system. Conversion of decimal numbers to their binary equivalents is quite straightforward. The decimal number is successively divided by 2 and the remainder is recorded (including 0) until all the digits have been exhausted.

Thus in the case of 29_{10} (the subscript denoting that it is to base 10); dividing successively by 2 and recording the remainder yields; —

29	14	7	3	1	0
1	0	1	1	1	

The required binary number is 11101. Note that the digits are produced in reverse order, i.e., the *least significant* digit first.

The above method works only for integers, ie., whole numbers. To convert fractions we must instead multiply successive digits by 2 and record the value of the digit (1 or 0) carried to the left of the decimal point and then discard it. In this case the highest order binary digit is produced first. Thus for the decimal fraction 0.78125 we get:—

	0.78125
	x2
1	1.56250
	x2
1	1.12500
	x2
0	0.25000
	x2
0	0.50000
	x2
1	1.0000000

i.e., $0.78125_{10} = 0.11001_2$ the required binary equivalent. Of course, not all decimal fractions will convert to exact binary numbers, for example the decimal fraction $1/3 = 0.01010101 \ldots = 0.\dot{0}\dot{1}_2$ recurring.

The addition and subtraction of binary numbers proceeds in exactly the same way as for decimal numbers except that we carry 1's to the left when the total in a column reaches 2, e.g.

$$\begin{array}{r} 110101 \\ +\quad 1101 \\ \hline 1000010 \end{array}$$

and, in paper calculations, we can use positive or negative numbers at will.

However, the principal use for the binary notation is with digital processors. These are generally organised to operate with binary numbers of fixed length, referred to as 'words'. The length used is usually a multiple of 8 bits. The single 8-bit word is referred to as a 'Byte'.

It is desirable that a single word should embody both the *sign* and the *magnitude* of the quantity which it represents. This is accomplished with the use of two's complement arithmetic. In this method, assuming a word length of 1 byte, zero is represented by 00000000. If we substract 1_{10} from 0 we get -1_{10}. Carrying out this operation on the binary zero, i.e.,

$$\begin{array}{r} 00000000 \\ -\ 00000001 \\ \hline 11111111 \end{array}$$

Thus -1_{10} is represented by 11111111, the left-most digit indicating the sign which is 1 for negative numbers and 0 for positive numbers. The range of numbers which can be represented will be from -128_{10} to $+127_{10}$, which in binary form is from 10000000 to 01111111. The sign automatically changes as we count through zero from —1 to +1. Logically, two's complement numbers are easy to handle. To form the negative of a number we simply invert each digit and add 1 e.g.,

$+22_{10} = 00010110_2$ and $-22_{10} = 11101010_2$

The use of two's complement representation has a number of advantages:—

i) The sign of the number is contained in the left-most digit.
ii) Zero is uniquely represented by 00000000 (assuming a 1-byte word).
iii) The odd or even character of a number is preserved in arithmetic operations which result in a change of sign. Thus, testing of the least significant digit to determine rounding can be used.
iv) It is not necessary to provide separate logic to perform subtraction since this operation can be performed by means of negation followed by addition. Addition of two's complement numbers is exactly the same as for ordinary binary numbers with the sign being automatically taken care of, for example:—

$$\begin{array}{rl} -12_{10} = & 11110100 \\ -\ 3_{10} = & 00000011 \\ \hline & 11110001 \quad = -15_{10} \end{array}$$

Thus far we have seen that a single eight-bit word may be used to represent either signed or unsigned quantities, it can however be used to represent many other things, depending upon the particular requirements of the programme.

Before we move away from numerical representations, we should consider other number systems which are used to ease the human task of handling what are basically multi-digit binary numbers. Human beings experience enough difficulty in reading and/or copying strings of decimal digits without error; these difficulties are increased manyfold if the decimal digits are replaced with binary numbers. For this reason *alternative* number bases have come into wide spread use as a short-hand for writing binary numbers.

These bases are the octal and hexadecimal (i.e. numbers to base 8 and 16 respectively). Both bases are related to a simple manner with the base 2; ie. $8 = 2^3$ and $16 = 2^4$, and this leads to a simple procedure for translating from one system to another. The **octal** system simply uses the symbols 0 through 7 of the decimal system whilst the **hexa-decimal** system requires additional symbols to represent the quantities 10 through 15; the letters A – F are used for this purpose.

A binary number can easily be translated to its Octal or Hexa-decimal equivalent simply by grouping the binary digits in groups of either 3 or 4 digits respectively, e.g.; – consider the binary number: 111010101101
In Octal:– 111,010,101,101 = 7255_8
In Hexa-decimal:– 1110,1010,1101 = EAD_{16}

Of the two systems, hexa-decimal seems to have become the more popular particularly with the advent of microprocessors. Both systems can be used to represent two's complement binary numbers without any modification. Considering an eight-bit binary number, the number range translates to:–

$+127_{10}$ to -128_{10}
$7F_{16}$ to 80_{16}
177_8 to 200_8
01111111_2 to 10000000_8

A final, numerical use of the eight-bit word is Binary-Coded Decimal (BCD). Since only four binary digits are required to represent the decimal digits 0 – 9, an eight-bit word can represent two decimal digits in BCD form. Thus a single word can represent the decimal number range 0 – 99.

The most important non-numerical use of the binary word is perhaps its application to representing the alpha-numeric (letter-and-number) character set used by most input/output devices such as teleprinters and video displays. Various codes have been used but probably none as widely as the ASCII. (American Standard Code for Information Interchange).

The code is in a number of distinct parts corresponding to upper and lower case alpha characters, numerals, punctuation and control characters. An eight-bit code is used, seven bits represent a letter or number and bit 8 is used as a parity check bit.

Parity checking is an elementary form of error detection. Dependent upon whether ODD or EVEN parity is required, an additional bit is added to the code so that the total number of 1's in any character generated will be odd or even. For example in the code word PO111101 where P represents the parity bit, for even parity P = O; whereas for odd parity P would be set to 1 (i.e., so that the total number of 1's shall be even – in this case = 6).

If a code is transmitted with a known parity, then a simple check at the receiving end can determine whether or not the received parity is what is expected. This simple system will only detect an odd number of errors in a given word; an even number of bit changes will leave the parity of the word *unchanged* and therefore no error would be detected even though it existed.

Returning to the ASCII code, it is valuable to note that the numbers 0 – 9 are represented by 30_{16} to 39_{16}. Hence within the processor it is fairly easy to convert these decimal digits directly to binary and vice-versa. In fact, if one writes out the binary equivalent of 30_{16} to 39_{16} it is seen that the conversion to binary is simply a matter of discarding the four most significant bits.

The computer word need not be used to represent intelligence in any of the ways discussed above. It can be used simply as *eight separate* input/output lines. For example suppose we have eight transducers, e.g. switches, photo-electric beams etc, and we wish to record the number of times that each is operated. *Each* of these separate devices can be connected to a *single* input port of our processor via a suitable interface, so that when one of the devices is actuated an appropriate bit in the input word is set to logical value 1. In our programme we arrange to sample the port concerned periodically and read into the "A" register one eight-bit word.

The contents of the A register are first tested to see whether they are zero. If the A register is non-zero it is then rotated left eight times. After each rotation the carry bit is examined; if it is 1 then the memory location associated with the input represented by the bit which has just been rotated into the carry bit is incremented by one. At the end of the input cycle the input buffers for the port are *reset,* so that on the next cycle only newly set inputs will be registered. The rate at which the port is sampled must be determined by the maximum rate at which the input signals can arise.

In a similar way a single computer word can be used to selectively *output* eight separate signals, for example to actuate solenoids. For example, by setting the A register to the value 08_{16}, bit number 3 only is set to 1 and this could be output via a suitable interface to actuate a solenoid connected to this bit of the output word.

In the opposite direction we can string together two or more words in order to handle numbers of greater magnitude than can be handled by a single eight-bit word. Many high level language compilers, notably FORTRAN, permit the use of two words to represent integers, thus extending the number range to -32768_{10} – $+32767_{10}$. Integer arithmetic has its limitations especially when we wish to handle numbers which have a fractional part – referred to in FORTRAN as REAL numbers. The commonest way of handling these numbers is by means of floating point representation in which all numbers are scaled so that they are of the form $a \times 2^b$ where a is always less than 1, and the exponent b is always an integer.

Multiplication (division) is achieved by multiplication (division) of the mantissa a and addition (sub traction) of the characteristic b. For addition and subtraction the characteristics (b) must be equal and the mantissas must be adjusted accordingly. Single precision is normally carried out with double word characteristics and mantissas; double precision arithmetic is also provided using four words for each.

We have discussed a few of the ways in which a binary word may be used to represent intelligence. The treatment is by no means exhaustive but it is hoped that it will serve as an introduction to the subject.

GETTING IT TOGETHER

Build your own assembler

Mike Banahan

PART 2

The last article took a distant look at assembly language and gave a general explanation of what an assembler is used for. For those of stout heart and a determined will, this time all will be revealed and the exotic mysteries of the innards of an assembler will be laid bare. Well, enough to be getting on with, anyway!

TWO PASS OR NOT TWO PASS?

Most assemblers are of the "two pass" variety, that is to say they have to read the source (input) text twice or three times, depending on whether or not the system they run on allows them to produce a listing of the source code and a binary (object code) output at the same time. It is possible to produce an assembler that only reads the source code once, but the restraints that have to be placed on the programmer to allow this to happen remove one of the advantages that assembly language is supposed to provide, namely making the messy business of machine code programming easier for the programmer. Why, you may ask, is this so? Well, that's an easy one. It has to be done to remove what are called FORWARD REFERENCES. These naughty items are to be found whenever this sort of situation crops up:

```
   JUMPTO LABEL
   CODE
   MORE CODE
LABEL:
```

On the first pass, the assembler gets to JUMPTO but doesn't know anything about LABEL. It isn't able to fill in the bit of the JUMP instruction that tells the processor where to go, so it doesn't bother and continues reading the input. It gets to LABEL eventually and enters LABEL into its SYMBOL TABLE together with the address at that point. On the second pass, when it reaches the JUMP instruction, it digs around in the symbol table looking for the address associated with LABEL. Having found it, it can then put the right information into the instruction.

Whilst we're talking about the symbol table it's worth noting that another type of entry is usually allowed for. This is the VARIABLE type of entry. Variables are provided for the convenience of the programmer and are, as their name implies, allowed to change value. Typically, a variable could be declared by a statement of the form:

```
   SPACE=   40
```

In the assembler to be described later, that would allocate the value 40 (octal) to the symbol SPACE, allowing the programmer simply to specify SPACE when requiring the value of the ASCII character " " . This is quite handy, especially from the point of view of somebody who wants to understand the program when either coming back to a program written some time ago or to one written by someone else. Imagine the internal dialogue "Compare the accumulator with 40? What the *****for?" or instead, "Oh yes, looking for a space character".

Of course, the vicious minded can just as easily use the same technique to reduce the intelligibility of a program, (opaque programming is a field worthy of a separate article) but that's a different matter, and the best of luck to anybody who wants to try. How would you like to try debugging some code which had SEMICOLON set equal to the value of a space? Try it sometime.

Another step in the same direction is to allow mathematical operations on these IDENTIFIERS. Maybe there's a table of data that needs to be set up at run time and you want to know how long it is. It's very handy to be able to write the code like this:

```
   BOTTOMLABEL:
 TABLE
 OF
 DATA

   TOPLABEL:
LENGTH=TOPLABEL-BOTTOMLABEL
```

The degree of complexity allowed in expressions of that sort varies between assemblers but it's unusual not to find at least addition, subtraction, multiplication and division.

The important distinction between labels and variables is that whilst a variable can take a number of values at different points in the assembly, (VARIABLE= VARIABLE*2 is quite legal) a label must have only one value, the value of the address at which the label was declared. Attempting to declare a variable twice is an error and must be flagged as such. A flowchart for the process of dealing with identifiers is to be found at the end of the article.

The real job of the assembler is of course to produce machine code output. To do this the assembler must read through the source text and decide from the characters it is receiving what it should be doing. The difficulty of the task depends greatly on the SYNTAX of the language being assembled. Syntax is not, as you might think, something the Inland Revenue claim from the morally suspect, but is a word borrowed from those arty types, the linguists. It is a term used to describe the order of words in a language. Think about this: The operation of adding register B to register A is specified absolutely (as far as the computer is concerned) by one particular combination of noughts and ones in the instruction. The assembler syntax, on the other hand, could be any of these:

```
ADD     A, B
ADD     B, A
A=A+B
A, B    ADD
```

and so on. None of them is particularly wrong, but some are easier than others for the assembler to decode. The usual procedure and easiest to deal with is to have the opcode first, followed by the operands, if any: as in the first example above.

The number of ways that can be chosen to check the syntax and generate the output code are almost endless, and there just isn't a best answer which fits all machines and any syntax. There are, though, some points worth thinking about if you decide to design your own syntax.

1) It doesn't matter how you process it as long as correct code is produced at the end of assembly, unless your procedure is so slow that results never appear.

2) Try and make the syntax fairly consistent and avoid too many "special cases" - the assembler is meant to make it easy for the programmer and should deal with most of the "funnies" itself. Example:

```
ADD     A, B
```

If A could legally be either a register or a variable or label, confusion would arise as to what is meant by the instruction above. Is A being added to B (two registers), or is the value *represented* by the symbol A implied? It's up to the person writing the assembler to implement a solution to the problem. Please don't skirt round it by insisting on a nasty rule such as "no labels may be used which could be mistaken for a register". At a pinch it might be justifiable to insist on a different opcode, perhaps ADDR for add registers and ADDC for add constant, but that isn't really helping the programmer who uses the assembler. A bit of extra effort by the assembler writer can make things a lot easier for the assembler user.

3) Avoid like the plague (this is a direct contradiction of 2 above) syntax dependent opcodes. All right then, at least try to reduce them. They can become the bane of your life!

Take for example a beautiful, straightforward opcode like HALT. Most machines have a few like that, and they are dead easy to deal with, just produce the code for halt and it's out of the way. The forthcoming Z-80 assembler contains some vile syntactic constructions: 112* < LABEL/3 > (IX) ← A takes a bit of sorting out, it isn't until you've reached the ← that you can even be sure that you're dealing with a LOAD instruction. It just makes the assembler easier to use later.

4) Write down a definition of your proposed syntax before you start and make sure that you understand it. Not that you're likely to produce one without ambiguities anyhow, but it helps to have something to work to.

5) Flatly reject ridiculous suggestions such as "labels either start in column 1 of the text or terminate with semicolon.". That sort of thing needs to be drowned at birth. A label is a label wherever it appears and must be easily identified as such. Any suggestions to the contrary should be met only with scorn.

MYTHOLOGY AND ANCIENT GREASE.

Both Helen of Troy and the Gadarene Swine were constructed of pretty much the same sort of parts - a handful of bones, some hair, a great deal more intestine than you would think, a large quantity of water and so on. The difference lay in the way they were fitted together. It's pretty much the same with assemblers.

A lot of the skeleton of an assembler is independent of the syntax to be processed, so once you've got it written, you can use it for the rest of time to come. The input/output routines, mathematical evaluation routines, table searching and other utility routines take up quite a lot of space and a lot of writing. Typically they may make up 60 per cent of the code in an assembler and are not trivial by any means. To help anybody interested, the flowcharts for some of these processes are included at the end of the article.

BUT HOW DO YOU DO IT?

That's a hard one. There isn't any "right" way of dealing with the problem, but an approach which seems to work in at least some of the cases is the one used in my Z-80 assembler.

The input text is read serially, character by character, into a space in store, until a TERMINATOR character is found. The group of characters just read is treated as the smallest "particle" of text that's of any interest and gets called an "atom". Leading spaces and comment are ignored, which allows the programmer to have some control over the layout of the source text.

By taking this path, the add instruction mentioned before (ADD A, B) becomes broken down into:

```
ADD
A,
B
```

From which it becomes evident that space and comma must be terminators. As soon as the ADD part has been read, the assembler goes to see if ADD is a valid opcode. It does this by searching a table of opcodes which also contains the address of where to process that opcode if it exists. Having found ADD in the table, the assembler jumps to the indicated address and continues processing from there. By calling for the next atoms as they are required and checking their legality in the particular context, the whole instruction can be decoded more easily than by attempting to "solve" the instruction all in one go.

The routine to deal with ADD needn't necessarily be specifically for the processing of ADD, but it depends on the processor that the code is being prodcued for whether or not other opcodes can be dealt with in the same place. It usually turns out to be the case that several instructions for a processor are very similar and can be processed by the same routine, with a considerable saving in size of the assembler. Other techniques are available to reduce the size of the assembler and the amount of work it does. It might prove to be possible to include information about the individual opcodes in a table of "permitted operands". In turn that table could be coded to allow a single routine deal with the majority of the opcodes. Which will be the best method depends on the instruction set of the particular processor you are working on.

Applying the ancient principle of never doing more work than is necessary (unless it seriously affects your fee), the next principle to be employed is that beloved by educational psychologists, "Learn By Example". Instead of reading any more hard - to - understand words, wait 'till next time round and have a look at some pictures, in this instance the Z-80 assembler in the flesh. It's written in its own syntax which should help to explain the syntax better than any formal introduction, but you get one of those too. A listing is worth a thousand articles.

SNEAK PREVIEW – SOME FLOWCHARTS

First a few words of explanation about these flowcharts. Although there will be an individual note on each one it's only fair to point out now that they are not exactly the same as the ones from which the Z-80 assembler was written. Since the originals went the way of all documentation, either lost or never written at all, these are lifted from a cross-assembler written on a PDP-11. They have the great advantage of being considerably more likely to be correct because of this! They should be quite easy to code for almost any machine.

GETATM. – GET next ATOM

This routine reads the input code and ignores leading tabs, spaces and comment. It assumes that a line of text has been read into a buffer in core and that the line terminates with carriage return. The reading of a line is dealt with by a separate routine. Two pointers are used to store the start and finish addresses of the atom. The pointers are assumed to point at the previous atom when GETATM is entered.

EVALU AND EVAL.

EVALU deals with mathematical expressions. This is not much like the corresponding routine in the Z-80 assembler which is horribly wrong. EVALU uses three stacks and another routine EVAL. EVAL can call EVALU recursively so EVALU has to consist of "pure code". EVAL produces a number from a simple expression, say VALUE when VALUE has been defined as a label or variable, and simple numbers, say 127 for example. EVALU deals with things like 1*2/3. When EVALU finds open brackets "<" it assumes that the thing inside the brackets is just a simple expression. Since EVAL can't deal with anything other than a simple expression, it uses EVALU to evaluate whatever is *inside* the brackets. Hence the recursion. Normal algebraic notation is used, *and / having precedence over + and –. Bracketed expressions take overall precedence. Try and make sure that you understand the use of the stacks because it isn't simple. Remember that EVALU expects to find the bottom limit of the operand and operator stacks on the subroutine return address stack - these addresses must be there whenever EVALU is called. They are normally set up by calling EVALU via a simple initialising routine.

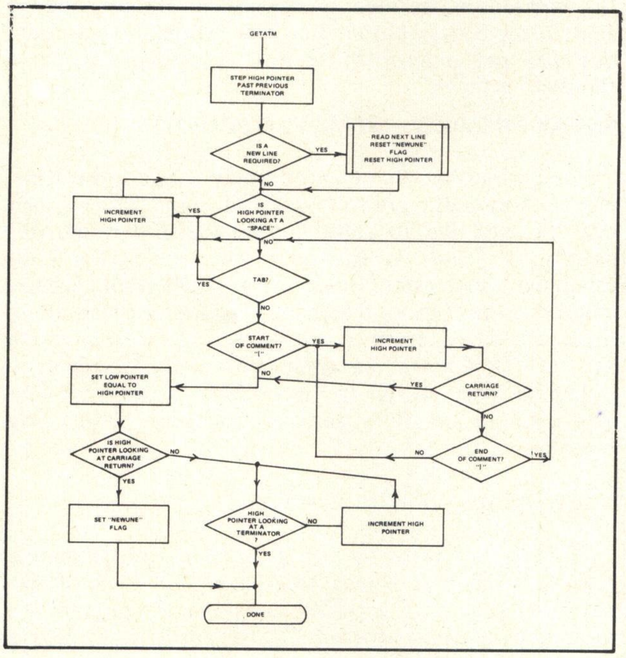
LABEL.

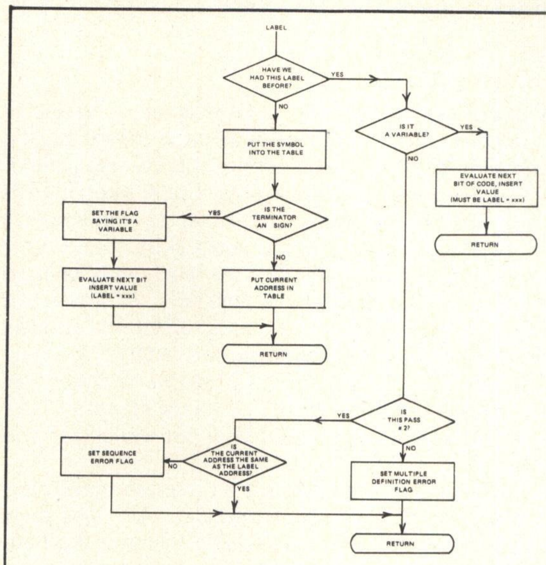
Label comes straight from the Z-80 assembler and describes the process of dealing with labels and variables. It is pretty simple. One or two things need a little explanation, though. An entry in the symbol table consists of a string of text (the characters in the label) and the value associated with that label. As the characters only need 7 bits to define them and up to six characters are allowed, there are six bits spare in a label. (In an 8 bit processor). These bits are used to flag things, notably whether or not the label is really a variable, if the label has been multiply defined, and if a sequence error has occured. Sequence errors are often caused by this sort of mistake:

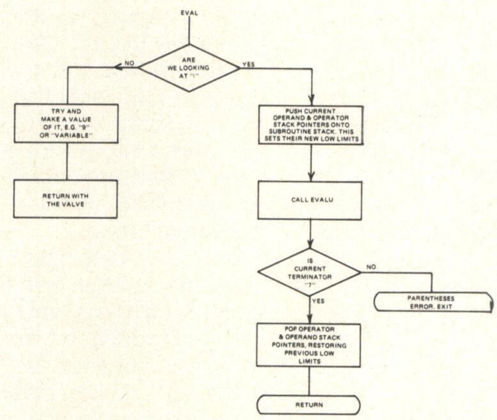
BLKW LABEL 1–LABEL 2; ALLOCATE SOME STORAGE
LABEL 1:

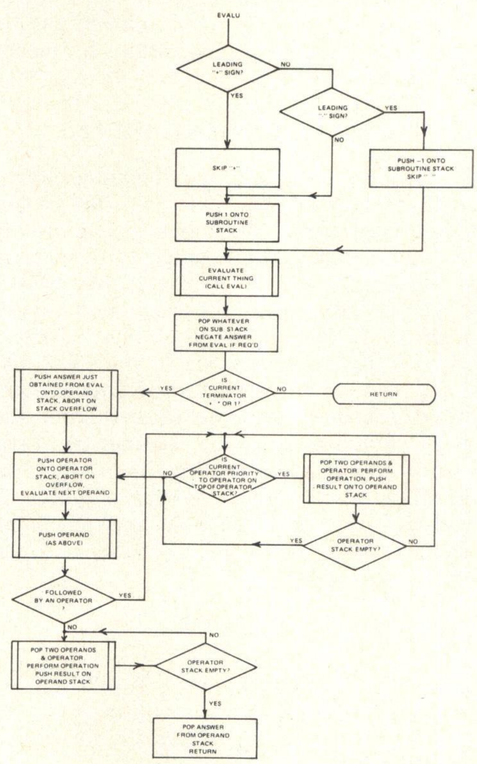
LABEL 2:

On the first pass, LABEL1 and LABEL2 are undefined so the storage block allocated by the .BLKW directive will not be the correct length. On the second pass, the labels will be defined and a different amount of space will be produced by .BLKW. Lo and behold, when LABEL1 is reached, the address counter will not be the same as it was last time. The error is known as a sequence or phasing error.









IN PRAISE OF THE PDP~11

PART – 2 Mike Lord

PROCESSOR ARCHITECTURE

Word Size

The PDP-11 is basically a 16 bit machine, in that the data bus is 16 bits wide, the processor can handle 16 bit data *words*, the 'natural' addressing range (without recourse to Memory Management schemes) is defined by 16 bits, and instructions use 16 bit words.

However, for many applications, the basic unit of data handled by a computer system is less than 16 bits, and is typically an eight bit *byte* as in, for example, text handling and BASIC interpreters. In these cases a 16 bit data word would be a disadvantage, so to overcome this problem the PDP-11 has been designed to handle 8 bit data bytes as easily as 16 bit words.

To achieve this the majority of the data manipulation instructions are available in two forms; one which uses 16 bit operands, and a second form which operates on 8 bit bytes. Also, memory space is addressed in terms of bytes, or half-words, using the conventions.

An *even* address refers to a complete word if the instruction defines a 16 bit operation; or to the least significant 8 bits of that word if the instruction defines a byte operand.

Use of an *odd* address with a 'byte' operation refers to the most significant 8 bits of the word.

An odd address *cannot* be used with an instruction which has a 16 bit operand.

The last few years have seen much argument about the relative merits of 8 and 16 bit architectures, particularly from microprocessor manufacturers (it is interesting to note how the arguments in favour of 16 bit words have gained strength in parallel with the manufacturers' expertise in producing 16 bit microprocessor chips), but it is worth reviewing the main points to be considered when making a choice:

Program Speed. Theoretically a 16 bit machine will execute an algorithm faster than an 8 bit machine because it can transfer information between CPU and store faster. For a given cycle time a 16 bit data bus can carry twice as much information as an 8 bit one. And even if the data you are processing is based on 8 bit bytes, don't forget that instructions, which are often 16 or more bits long (when you include the addressing portion) have to be transferred over the bus as well. Also, most programs involve a fair amount of address manipulation, requiring movement of 16 bit address words over the data bus, and often arithmetic calculation of addresses by the program, which is performed much faster by a 16 bit machine.

Memory Size. The number of bits of memory required for a particular application is determined by the amount needed to hold the data, which is not really affected by the word size, and the amount required to hold the program. The amount of memory the program takes depends to a large extent on the programmer's skill and the amount of time he spends. (One programmer I know says he can reduce the size of any program by 10% – I sometimes wonder for how many iterations). And in practice it is usually possible to trade speed of execution for memory size. The 'efficiency' of the instruction set will also have an effect on the memory spare required, but unfortunately the 'efficiency' of a particular machine language is almost impossible to define, it depends too much on the particular problem being programmed and the algorithm used. The only sure statement is that there is as much difference between the 'efficiency' of different 8 bit machines as there is between 8 and 16 bit architectures.

Ease of Use. If you are programming in a high level language then the fundamental machine characteristics are hidden, and it really makes no difference to you whether you are using an 8 or a 16 bit machine. However, for those condemmed to work in the murky depths of Assembly Language, the quirks of a particular machine's instruction set become painfully obvious, and a messy instruction set must surely prolong the time taken to write a program and complicate debugging. While, again, there is as much difference between different 8 bit machines as there is between 8 and 16 bit architectures, it seems reasonable that designers of a 16 bit machine would have a better chance of devising a powerful but simple to understand instruction set than those limited to the constraints of an 8 bit word.

Cost. As 90 – 95% of the cost of a computer system is in the memory and peripherals, any increase in cost of the central processor caused by a change from 8 bits to 16 bits is negligible.

Summarising, by designing a 16 bit machine which can also easily handle 8 bit data bytes, the PDP-11 designers have achieved the best of both worlds.

CPU General Registers

In addition to the main memory, all modern computers also have a number of registers associated with the CPU for holding data and addresses in frequent use. There are two main reasons for doing this;

As there are relatively few of them, and as they can be physically located close to the CPU, high speed devices can be used to give shorter access times than would be economic for main memory. Thus the overall system speed can be increased by holding the more frequently used words in the registers rather than in main memory.

The registers can be referenced by far fewer 'address' bits than can a word in main memory ; again because there are far fewer of them. This means that instructions which reference registers can be shorter (less bits) than those which reference words in main memory, so a significant saving in overall program memory size can be obtained by keeping the most frequently used information in registers. (Speed is also improved as shorter instructions mean fewer bits to be transferred over the system bus).

In general, registers may be used as temporary storage for variables, as 'accumulators' to hold interim results during a calculation, to hold address offsets or bases for indexing, as stack pointers and as a program counter. Some machine designers seem to delight in providing a selection of different types of register, each governed by a different set of instructions. For example the 6800 has 'A' & 'B' accumulators, with slightly different functions, also an Index Register, a Stack Pointer and a Program Counter.

The PDP-11 has 8 general purpose 16 bit registers (RO – R7), the only restriction on their use being that one (R7) is also used as the Program Counter and another (R6) as the system Stack Pointer. All of the fundamental data manipulation instructions can be applied to any of the 8 registers. (Rotating the Program Counter left one bit may not be required often, but it is possible!). This gives a consistency which makes the whole instruction set simpler to understand while keeping it more powerful than most. For example, to move the stack pointer by 6 locations, the PDP-11 user can use the single instruction;

```
      ADD#6,R6      (R6 is the stack pointer)
or    SUB#6,R6
```

While to copy the current value of the Program Counter (sometimes useful when writing position independent code) then;

```
MOV R7,TEMP (R7 is the PC. TEMP a
                 temporary storage location).
```

does the trick simply.

Although R6 is designated as the system stack pointer, any of the other registers R0 – R5 may also be used as another stack pointer. Alternatively, they may be used as general purpose accumulators, as address index registers, or as address pointers.

INSTRUCTION SET

Two Operand Instructions

Unlike most mini's and micro's, the PDP-11 has a true '2 Address' instruction set (see Table 1). That is, in two operand instructions such as;

ADD A to B
COMPARE A with B

there is no restriction on where the operands A and B may be stored. They can be in registers or main memory, and any addressing mode may be used to refer to either. Thus the contents of a register may be added to a memory location as easily as one register to another, or even one memory location to another. All of which makes machine language programming much easier than with CPU's such as, say, the 8080 which requires that the 'B' in 'ADD A to B' is located in one of the registers.

TABLE 1 PDP–11 2–OPERAND INSTRUCTIONS

MOV	Move A to B
CMP	Compare A with B (without altering A or B)
ADD	Add A to B
SUB	Subtract A from B
BIT	Bit Test (Logical AND of A and B)
BIC	Bit Clear
BIS	Bit Set (Logical OR of A and B)

All of which take the form;

15 – 12	11 – 6	5 – 0
Op Code	S	D

Where S and D fields define the source (A) and destination (B) locations of the data. All except ADD and SUB can be specified to operate on bytes (e.g. MOVB) as well as whole words (MOV).

This symmetry simplifies the instruction set (and so reduces the burden on the programmer's personal memory) as, for example, the single PDP-11 'MOV' instruction is all that is required to do the equivalent of the 6800's LDA (Load Accumulator from Memory), STA (Store Accumulator to Memory), and TAB, TBA (Transfer between accumulators), not to mention the LDX, LDS, STX, STS, TAP, TPA, TSX, TXS, PSH and PUL instructions.

Single Operand Instructions

These are shown in Table 2, and again can operate on data in main memory as well as on data in the registers. Except for SWAB, all single operand instructions can operate on either bytes or on 16 bit words. On a historical note, the PDP-11 was the first mini to provide the TST instruction which examines data and sets status flags accordingly (positive or negative, zero or nonzero) without disturbing the data. This instruction is particularly useful for examining the status of peripheral device control & status registers.

TABLE 2 PDP–11 SINGLE OPERAND INSTRUCTIONS

CLR	Clear
COM	Complement (1's)
INC	Increment (by 1)
DEC	Decrement (by 1)
NEG	Negate (2's complement)
TST	Test
ROR	Rotate right
ROL	Rotate left
ASR	Arithmetic shift right
ASL	Arithmetic shift left
SWAB	Swap bytes (in a word)
ADC	Add carry
SBC	Subtract carry

They are of the form;

15 – 6	5 – 0
Op Code	D

Where D defines the location of the data to be operated on.

Branch Instructions

One of the features of modern CPU design which was pioneered by the PDP-11 is the concept of 'status bits' which hold information about the last data manipulation type instruction to be executed (i.e. whether it gave a positive, negative or zero result, or whether it resulted in an arithmetic overflow or set the Carry bit). This status information can then be tested by 'conditional branch' instructions which can alter the program flow if the specified conditions are met. Thus the program sequence;

```
. . . . . . . . . . . .
CMP A, B
BLT PROG1
BGT PROG2
. . . . . . . . . . . .
```

branches to PROG1 if A was less than B, to PROG2 if A was greater than B, and continues without branching if A equals B.

This will, of course, be familiar to users of the 6800 and similar CPU's, as will the PDP-11's use of relative addressing (± 127 words range) for the branch instruction, however the point to bear in mind is that it was the PDP-11 that introduced this approach.

Addressing Modes

The way in which the PDP-11 specifies the addresses of memory locations and memory is, in the author's opinion, unequalled in elegance and power.

An instruction set should allow the programmer to specify addresses in a variety of ways, according to the natural requirements of the program. Thus data might be in a register or in main memory, on the top of a stack or a number of words below the top of stack. Data or subroutine addresses might also be held in an array, in which case indexed addressing is needed, or might be referenced via an array of pointers. Also, programs can be simplified if 'immediate' data is allowed (data word immediately following the instruction), and some programming problems are best solved by specifying memory addresses relatively, while others demand absolute addressing. Further, some applications require the program to search through a list of data words or bytes in memory, and others are made easier if more than one stack is available.

In most CPU designs these diverse requirements are met by defining specialised types of instruction to cater for the most common functions, and leaving the programmer to write his way round the missing ones. The PDP-11, however, solves the problem in a more general way.

As mentioned previously, the single and double operand types of instruction contain six bits to specify each address. Three of these bits are used to define one of the eight registers (R0 – R7), and the other three bits determine one of eight address modes. These modes are;

Mode 0, Register; R

The data is contained in one of the registers R0 – R7.
e.g. CLR R4 clears the contents of Register 4.

Mode 1, Register Deferred; (R)

The register contains the address of the required memory location. e.g. if R4 contains '1234', then;

CLRB (R4) clears (byte) location 1234
JMP (R4) causes a program jump to location 1234

Mode 2, Auto Increment; (R)+

As Mode 1, except that the contents of the register are incremented after they have been used as an address. The increment is by 1 if the associated instruction operates on a byte, by 2 if it operates on a 16 bit word.

Mode 3, Auto Increment Deferred; @ (R)+

As mode 2 except that the contents of the register are not the required address itself, but specify a location which contains the desired address. e.g. if R4 contains '1234'. and location 1234 contains '4567' then

CLRB @ (R4)+ Clears location 4567 (one byte)

Mode 4, Auto Decrement; –(R)

As mode 1, except that the contents of the register are decremented before they are used as an address. e.g. if R4 contains '1234'

CLRB –(R4) changes R4 content to '1233', and clears the byte at location 1233.

Mode 5, Auto Decrement Deferred; @–(R)

As mode 4 except that the contents of the register are a pointer to a location containing the desired address.

Mode 6, Indexed; X(R)

Where X is a 16 bit integer. In this case the value of X is added to the contents of R to give the address. e.g. if R4 contains '1234', then

CLRB 20(R4) clears location 1254

Mode 7, Index Deferred; @X(R)

As mode 6 except that the address obtained by adding X to the contents of R is a pointer to the location which contains the desired address.

The author can still remember how baffled he was on first reading about these addressing modes. Why all the trouble? What would all these modes be used for? But with some thought and a lot of experimentation things began to fall into place.

Take for example the auto increment mode (R)+. Since the program counter R7 is a valid register, then;

```
.............
MOV (R7)+,R0
#1234   (#1234 is an integer constant)
.............
```

moves the data 1234 to register 0, and increments the Program Counter to point to the instruction following the data word #1234. This gives us a Load Immediate Data #1234 to R0 instruction, and in practice the PDP-11 assembler allows you to write the above sequence as

MOV #1234, R0

Similarly, by using Mode 3 ; @ (R)+ with R7, the word following the instruction is treated as an absolute address;

```
....................
CLR   @ (R7)+
#1234
....................
```

clears the contents of the word at memory location 1234, and can be written in assembly language as

CLR 1234

Using the Index mode 6 with the Program Counter gives relative addressing;

CLR 1234(R7)

since the value 1234 is added to the current value of the Program Counter to give the effective address, while the Index Deferred mode gives indirect relative addressing;

CLR @1234(R7)

The Auto Decrement mode allows you to use any of the registers as a stack pointer; thus if R2 is pointing to the current top of a stack, then

MOV R0, – (R2)

will decrement R2 to point at the new top of the stack, and store the contents of R0 in that position, effectively a PUSH to the stack defined by R2. (PDP-11 stacks work from high addresses downwards, thus the 'top' of a stack is actually the lowest numbered address). Data is retrieved (pulled) from the stack by using the Auto Increment mode;

MOV (R2)+,R0

To access an item which is on the stack but which is not at the top, then the Indexed mode is useful, e.g. if R2 is being used as a stack pointer, then

MOV 2(R2),R0

copies the second word on the stack into R0.

The auto increment and auto decrement modes are also useful when moving blocks of data or when examining strings. The program below shows how two areas or memory, treated as two 1024 byte strings, can be compared. The strings start at locations STRINGA and STRINGB;

```
        MOV #STRINGA,RO   ; point RO @ start of
                            first string.
        MOV #STRINGB,R1   ; point R1 @ start of
                            second string.
LOOP:   CMPB (R0)+,(R1)+  ; compare bytes then
                            increment pointers.
        BNE NOTEQUAL      ; end comparison if a
                            mismatch.
        CMP R1, #STRINGA+1024 ; end of string?
        BLT LOOP
        .................
        .................
```

POSTAMBLE

If these articles have not conveyed the impression of a machine which is both powerful and easy to use, then the fault lies with the author, not with the PDP-11. Although he has never worked for DEC, either directly or indirectly, he did have the pleasure of using their machines a few years ago, and has since found his enthusiasm shared by many PDP-11 users.

Readers may care to note that a PDP-11 based system is sold by Heathkit, while Rapid Recall have now been appointed distributors for LSI-11 boards. Also, LSI-11's may now sometimes be found on the 'surplus' computer market for not unreasonable prices.

ADVANCED INTELLIGENCE!

Bill Davy

Howard Kornstein of Intel recently gave a presentation in London of Intel's new 16 bit microprocessor, the 8086. Having been there, and also having spent the last two years working on another 16 bit microprocessor I should like to report my first impressions of what is undoubtedly a very important new arrival on the 16 bit microprocessor scene.

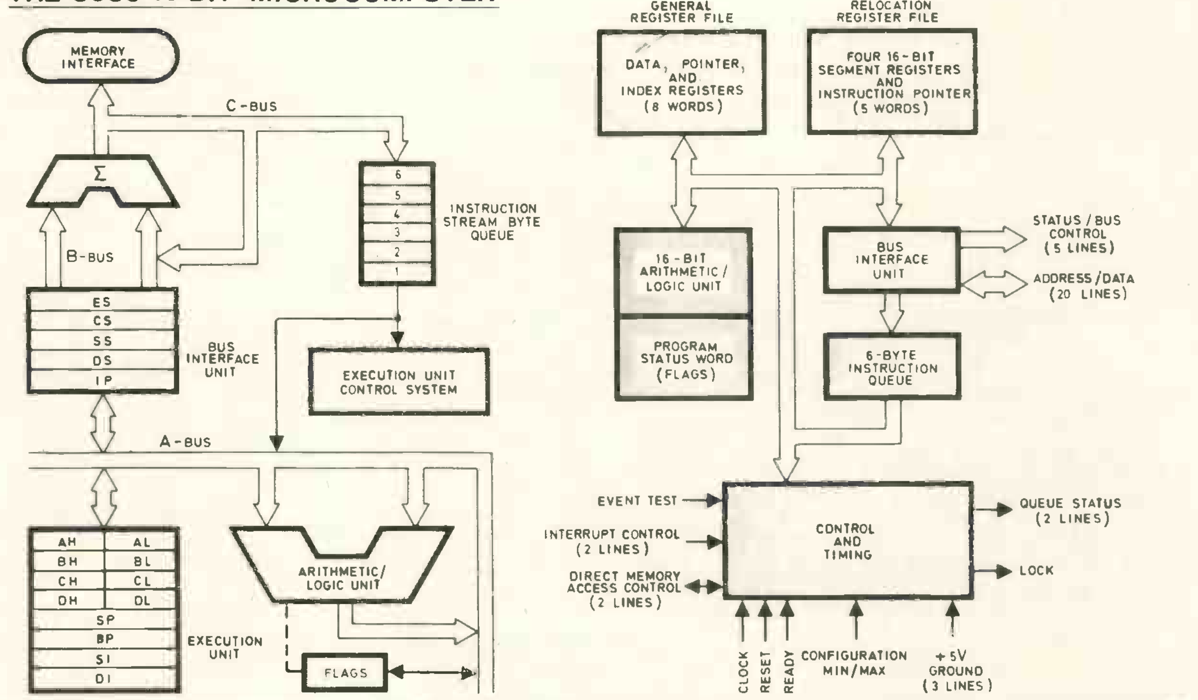
My first impression, and it is still with me, is that this is a superb processor. It is not particularly expensive; Rapid Recall (very generously) raffled a development kit costing £250. From this you will see that the processor and its support chips actually exist; as Howard Kornstein said, the 8086 is implemented on silicon, not paper!

Travelling toward miniland

The processor has many features not previously seen in microprocessors, though many are known in minicomputers. The processor has a 20 bit address bus and a 16 bit data bus. This would appear to give the processor access to 1 mega-word of memory but this is not the case. In order to be able to access a single byte of memory as well as words the least significant address line and another control line ($\overline{BHE}$/S7) are used to signal whether a word or odd or even byte is being accessed. The processor therefore only has access to 1 megabyte of memory; it is rumoured that Intel will give an 8086 away with every megabyte of memory ordered.

In order to use such a large address space with a 16 bit processor some memory address translation is required. In the 8086 this is achieved automatically by adding one of the four segment base addresses to the program produced address. As the segment base add-

THE 8086 16·BIT MICROCOMPUTER



resses are 20 bits wide (though they can only point to a 16 byte boundary as the 4 least significant bits are 0) this produces a 20 bit address. As an example of this, the sum of the instruction pointer and the code segment base address register is used to access instructions from memory. It is worth pointing out that the 8086 bus is compatible with the MULTIBUS used by Intel single board computers.

FIFO

Another advanced feature of this processor is the use of a fetch ahead first-in-first-out (FIFO) queue for instruction bytes. This means that instruction bytes are fetched from memory and placed in a queue waiting for execution. As a result of this there is no time wasted by locating instructions off word boundaries. This feature will also significantly increase instruction throughput in a system where several processors share the same bus. The queue is automatically cleared on any branch instruction.

Even with multiplexed data and address buses there is a problem with providing bus control signals which are economic for small systems and sufficiently powerful for large systems. Intel have used an elegant solution to this problem which provides the best of both worlds and keeps the pin count down to 40. An input to the processor called $MN/\overline{MX}$ determines the signals which appear on 8 other pins. When $MN/\overline{MX}$ is strapped to Vcc the 8 pins provide decoded control signals suitable for minimum systems, and when $MN/\overline{MX}$ is strapped to ground the 8 pins provide encoded control signals suitable for maximum systems.

This is hardly a comprehensive introduction to the hardware of the 8086 but it does give an idea of the many novel and exciting features available. The software features provided for the Intel 8086 are a little bit less spectacular but the none the less interesting.

Target instruction set

Howard Kornstein emphasised the suitability of the instruction set as a target language for compilation, and Intel already have a PL/M86 compiler ready. They intend to release a compiler for ANSI Fortran IV next year, although there was no mention of Pascal or Basic. A high level assembler is available (ASM86), as is a utility (CONV86) for converting 8080 assembly code to 8086 assembly code.

The register arrangement is a logical extension of that of the 8080's, which it includes as a subset. In general all addressing modes (and therefore all registers) can be used with all instructions where meaningful, although there are exceptions. As an example, the multiply and divide instructions require one operand to be in the A register. The processor has signed and unsigned, word and byte multiply and divide instructions. In general instructions exist in word and byte versions.

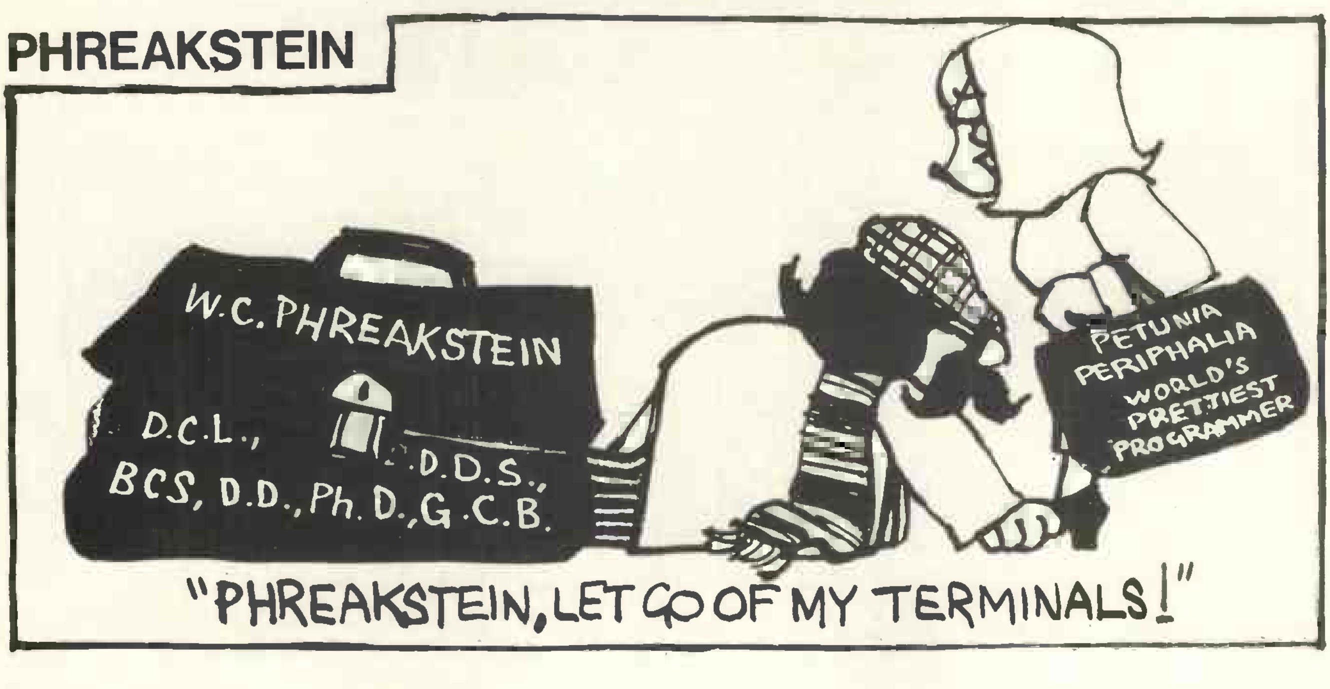
Felicitous Facilities

Interrupts are handled by a vectored single priority system, but even in this there are useful software twists. There is a status bit which, when set causes an interrupt to occur after each instruction (but not in the interrupt handling routine of course); this facility greatly facilitates the writing of a software trace program. Attempted division by zero also causes an interrupt, and there is a one byte instruction which causes an interrupt if the overflow status bit is set. In addition there is a one byte software interrupt instruction, which as those of you who are familiar with the Motorola 6800 will know is a useful way of inserting breakpoints into a program.

There are several string handling primitives which operate on words or bytes. They can be repeated until a count register (CX) becomes zero. If the primitive operation being repeated is an arithmetic one (compare or subtract for example) then the repetition will also cease when the Z flag reaches the desired state. The direction along which the strings are traversed is determined by the state of a user definable status bit.

The addressing modes are flexible and therefore powerful, although their power can be hard to utilise when compiling a high level language. It is for example possible to directly address the byte whose address is given by the sum of the base register (BX), the source index register (SI) and a 16 bit displacement specified in the instruction. If any deeper addressing is required it is possible to place the effective address of an operand into the extra data segment base address register. I am quite sure the available addressing modes will meet most good programming requirements.

In conclusion, it has to be said that the 8086 will amply repay the effort that is required to get to know it well. Its great software power combined with relatively simple hardware is sure to make it popular device for professional and amateur users alike.



The Soft Facade

G. J. Flanagan

One of the problems that any user of a microprocessor faces is that many systems have very little in the way of indication as to what is happening. To construct a workable system, various manufacturers have written and supplied short programmes in read only memory to allow the user to create and modify programmes. Some of the systems are designed to work with simple hexadecimal displays, whilst others provide the correct output to drive teletype like devices. Providing that programmes are fairly small and the user's hand coding is efficient, then this is all that may be required by many users. Should the programme not run correctly, however, problems develop when the user attempts to find what has happened to his programme. He needs to know whether it is waiting in a loop or, due to incorrect coding, is executing data rather than programme.

Two recent articles (Ref 1, 2) have described "trace" features to allow users of the M6800 to map out programmes as they are run. These articles have extended the software interrupt facilities already provided in many of the manufacturer supplied read only memories so that after each instruction of the user's programme a display of all active registers in the M6800 is created on the control terminal.

The beauty of these articles was the simplicity of the additional hardware required to obtain this feature. Figure 1 shows the connections required to M6800 system as supplied by South West Technical Products. Using the above hardware this programme provides the user with a hexadecimal printout of his programme including all relevant addresses, as the programme is run. It also displays the result of executing each instruction, showing what happens to all active registers in the M6800. Thus all the information that might have been available via lights and switches on a mini computer *can now be printed* on the control device of the M6800.

A typical output is shown in Figure 2. On the assumption that this device is usually a VDU the programme puts out column headings, and then a set number of instructions of the user programme are printed, including the way in which they modify any active register before the programme halts so that the text may be read. On depressing any key on the control device the programme will execute a further page full of instructions of the user programme. The feature can be activated at any time by setting up the correct address for a non-maskable interrupt in the locations A006 A007 and depressing the trace switch. Lifting this switch again will allow the user programme to continue at full speed.

There are two limitations in the use of this feature. One is that the user programme is slowed by the output to the control device therefore checking of time dependent routines is impossible. The second limitation arises from the fact that information is made available after the execution of a particular instruction. It is therefore very difficult to print out the first instruction of any trace operation since the programme effectively prints out the active registers followed by the programme counter and instruction for the *next* instruction to be obeyed and information is not available as to whether the first instruction obeyed is a one, two or three byte instruction. Therefore on entering the trace feature printing of the first instruction is suppressed. The only trick used to generate this system is one in which the non-maskable interrupts of the M6800 are effectively switched off.

In performing the trace, one non-maskable interrupt pulse is generated for each instruction of the user programme. In order to access the instruction in memory so that it may be printed on the VDU additional NMI pulses will occur. However at this point in the trace programme the NMI routine pointer word is changed to point directly to an RTI (return from interrupt) instruction so that access to traceable memory from within the trace programme does not cause the already interrupted programme to be interrupted again other than momentarily whilst the active registers are pushed on to the stack and pulled off again by the non-maskable interrupt and its following RTI instruction.

The programme is entirely pre-locatable but for it to work using the logic of Figure 1 it must reside above 8000. In our own system it forms part of our standard read only memory operating system located at $DF00 upwards with the six words of random access memory required being located in the "Mikbug" RAM area at $A018 upwards. The programme is normally used with a SWTP 6800 kit and has been checked using both the Motorola "Mikbug" and also the "SWTBUG" firmware.

In conclusion, the developed programme has been of use on a number of occasions both to assist in the teaching of the micro-processor language and to find programme loops which otherwise require extensive use of the software interrupt facility to trace the actual error. With the trace programme in the read only memory these problems can now be simply solved at the flick of a switch.

REFERENCES

M6800 trace facility. J. Brydon. New Electronics 16th May 1978 P.18.

M6800 single instruction execution. B. J. Frost. New Electronics 7th March 1978 P.19.

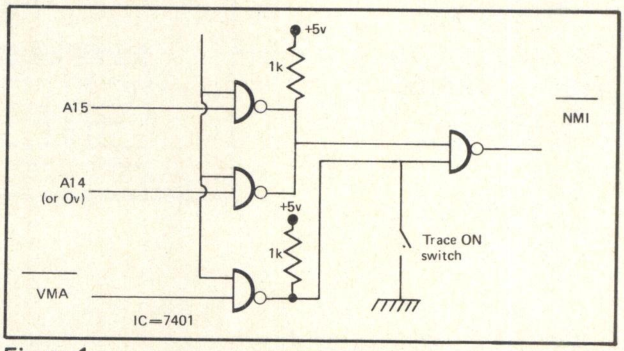
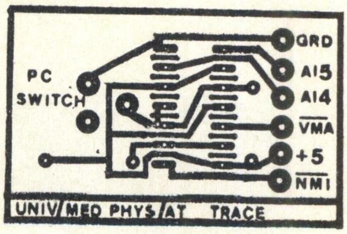


Figure 1



Printed Circuit Layout

```
$J 0000
 PC    INSTR.  HINZVC B  A   IX   SP      Print titles
               010100 B9 00 0000 A042     Set index register
0003 86 FF     011000 B9 FF 0000 A042     Load A with FF
0005 5F        010100 00 FF 0000 A042     Clear B
0006 01        010100 00 FF 0000 A042     No operation
0007 01        010100 00 FF 0000 A042     No operation
0008 08        010000 00 FF 0001 A042     Increment X register
0009 4A        011000 00 FE 0001 A042     Decrement A
000A 5C        010000 01 FE 0001 A042     Increment B
000B 36        010000 01 FE 0001 A041     Push A
000C 37        010000 01 FF 0001 A040     Push B
000D 32        010000 01 01 0001 A041     Pull A
000F 33        010000 FF 01 0001 A042     Pull B
000F 97 F0     010000 FE 01 0001 A042     Store A @ FØ
0011 7F 0007   010000 FE 01 0001 A042     Jump to LOC 7
0007 01                                   No operation.  Wair for key press
 PC    INSTR.  HINZVC B  A   IX   SP      Print titles
               010000 FF 01 0001 A042     Next pass through loop
0008 08        010000 FE 01 0002 A042
0009 4A        010100 FE 00 0002 A042
000A 5C        011000 FF 00 0002 A042
000B 36        011000 FF 00 0002 A041
000C 37        011000 FF 00 0002 A040
000D 32        011000 FF FF 0002 A041
000E 33        011000 00 FF 0002 A042
000F 97 F0     011000 00 FF 0002 A042
0011 7F 0007   011000 00 FF 0002 A042
0007 01        011000 00 FF 0002 A042     Pass 3
0008 08        011000 00 FF 0003 A042
0009 4A        011000 00 FF 0003 A042
000A 5C        010000 01 FE 0003 A042
000B 36
```

Figure 2

```
                          LIST
                    *
                          NAM "TRACE"
                    *
        DF00              ORG $DF00
                    *
DF00    CE DFC9  START    LDX #MES         PRINT HEADINGS
DF03    BD E07E           JSR $E07E
DF06    C6 0D             LDA B #13
DF08    BD DFAB           JSR SPACE        PRINT LEADING SPACES
DF0B    86 0E             LDA A #14
DF0D    B7 A01C           STA A COUNT      SET LINE COUNT
DF10    CE DF16           LDX #START1
DF13    FF A006           STX $A006        RESET INTERRUPT POINTER
                    *
DF16    BF A01A  START1   STS SP           SAVE USER S.P.
DF19    FE A01A           LDX SP
DF1C    08                INX
DF1D    E6 00             LDA B 0,X        GET CONDITION CODES
DF1F    59                ROL B
DF20    59                ROL B            TOP BITS NOT USED
DF21    86 06             LDA A #6
DF23    B7 A01D           STA A CNTR
DF26    4F       L1       CLR A
DF27    59                ROL B
DF28    49                ROL A
DF29    8B 30             ADD A #$30
DF2B    BD E1D1           JSR $E1D1        PRINT CONDITION CODES
DF2E    7A A01D           DEC CNTR
DF31    26 F3             BNE L1
DF33    C6 01             LDA B #1
DF35    8D 74             BSR SPACE        PRINT SPACE
DF37    08                INX
DF38    BD E0CA           JSR $E0CA        PRINT ACC B
DF3B    BD E0CA           JSR $E0CA        PRINT ACC A
DF3E    BD E0C8           JSR $E0C8        PRINT INDEX REG
DF41    FF A018           STX XTEP
DF44    86 07             LDA A #7         CORRECT STACK POINTER
DF46    BB A01B           ADD A SP+1
DF49    B7 A01B           STA A SP+1
DF4C    CE A01A           LDX #SP
DF4F    BD E0C8           JSR $E0C8        PRINT STACK POINTER
DF52    CE DFED           LDX #MES2
DF55    BD E07E           JSR $E07E        PRINT CR/LF
DF58    FE A018           LDX XTEP
DF5B    BD E0C8           JSR $E0C8        PRINT PROGRAMME COUNTER
DF5E    09                DEX
DF5F    09                DEX
DF60    EE 00             LDX 0,X
DF62    FF A018           STX XTEP

"TRACE"                                         ASM68 V1.1 PAGE 0002

DF65    CE DFC8           LDX #A2          PREVENT NMI'S TO GET INSTR.
DF68    FF A006           STX $A006
DF6B    FE A018           LDX XTEP
DF6E    A6 00             LDA A 0,X
DF70    36                PSH A
DF71    BD E0CA           JSR $E0CA        PRINT INSTRUCTION
DF74    FF A018           STX XTEP
DF77    C6 05             LDA B #5         ADDRESS HANDLING ROUTINE
DF79    32                PUL A
DF7A    81 8E             CMP A #$8E
DF7C    27 18             BEQ OUT1
DF7E    81 8C             CMP A #$8C
DF80    27 14             BEQ OUT1
DF82    81 CE             CMP A #$CE
DF84    27 10             BEQ OUT1
DF86    CE DFF0           LDX #TABST
DF89    84 F0             AND A #$F0
DF8B    27 05             BEQ OUT
DF8D    08       L2       INX
DF8E    80 10             SUB A #$10
DF90    26 FB             BNE L2
DF92    A6 00    OUT      LDA A 0,X
DF94    27 0C             BEQ S2
DF96    FE A018  OUT1     LDX XTEP
DF99    81 01             CMP A #$01
DF9B    26 09             BNE OUT3
DF9D    BD E0CA           JSR $E0CA        PRINT ADDRESS + SPACES
DFA0    C6 02             LDA B #2
DFA2    8D 07    S2       BSR SPACE
DFA4    20 0E             BRA CONT
DFA6    BD E0C8  OUT3     JSR $E0C8
DFA9    20 09             BRA CONT
DFAB    86 20    SPACE    LDA A #$20
DFAD    BD E1D1           JSR $E1D1
DFB0    5A                DEC B
DFB1    26 F8             BNE SPACE
DFB3    39                RTS
DFB4    CE DF16  CONT     LDX #START1      RESTORE NMI CAPABILITY
DFB7    FF A006           STX $A006
DFBA    7A A01C  A1       DEC COUNT        CHECK FOR END OF PAGE
DFBD    26 09             BNE A2
DFBF    CE DF00           LDX #START
DFC2    FF A006           STX $A006
DFC5    BD E1AC           JSR $E1AC        WAIT FOR KEYBOARD IF PAGE FULL.
DFC8    3B       A2       RTI
                 *
DFC9    0D       MES      FCB $0D,$0A
DFCB    20                FCC " PC   INSTR. HINZVC B  A   IX   SP"
DFED    0D       MES2     FCB $0D,$0A,$04
                 *
DFF0    00       TABST    FCB 0,0,1,0,0,0,1,2,1,1,1,2,1,1,1,2
                 *
        A018              ORG $A018
        0002     XTEP     RMB 2
        0002     SP       RMB 2
        0001     COUNT    RMB 1
        0001     CNTR     RMB 1
                 *
A048    0000              END
```

A FILLING ROUTINE

John Coll

FILL

This routine enables a block of memory to be filled with a specified byte.

The general syntax of the FILL command is FILL, (start addr) (end addr) (byte)

A few examples follow:

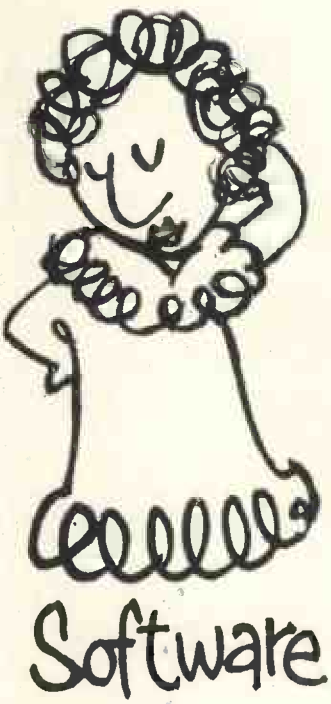
+++FILL 10   1000 BD
+++FILL 100 2000 3F

This routine is particularly useful for placing $3F in sections of memory to catch 'runaway' programs using the monitor SW1 re-entry routines.

```
                      NAM    FILL MEMORY WITH A CODE
7600                  ORG    $7600
                      OPT    PAG,NOG
                *
                *JOHN A. COLL  OUNDLE SCHOOL 8-6-78
                *
                *EQUATES
A002            BEGA    EQU    $A002
A004            ENDA    EQU    $A004
7103            WARMS   EQU    $7103
7118            PSTRNG  EQU    $7118
713F            GETHEX  EQU    $713F
7435            SKIPSP  EQU    $7435
7600 20 01      START   BRA    BEGIN
7602 01         VERSION FCB    1
7603 BD 74 35   BEGIN   JSR    SKIPSP
7606 BD 71 3F           JSR    GETHEX
7609 5D                 TST B
760A 27 28              BEQ    ERR
760C FF A0 02           STX    BEGA
760F BD 74 35           JSR    SKIPSP
7612 BD 71 3F           JSR    GETHEX
7615 5D                 TST B
7616 27 1C              BEQ    ERR
7618 FF A0 04           STX    ENDA
761B BD 74 35           JSR    SKIPSP
761E BD 71 3F           JSR    GETHEX
7621 FF 76 57           STX    XT
7624 B6 76 58           LDA A  XT+1
7627 FE A0 02           LDX    BEGA
762A A7 00      LOOP    STA A  X
762C BC A0 04           CPX    ENDA
762F 27 09              BEQ    FLEX
7631 08                 INX
7632 20 F6              BRA    LOOP
7634 CE 76 3D   ERR     LDX    #ERRMSG
7637 BD 71 18           JSR    PSTRNG
763A 7E 71 03   FLEX    JMP    WARMS
763D 49         ERRMSG  FCC    /Illegal address specified/
763E 6C 6C
7640 65 67
7642 61 6C
7644 20 61
7646 64 64
7648 72 65
764A 73 73
764C 20 73
764E 70 65
7650 63 69
7652 66 69
7654 65 64
7656 04                 FCB    4
7657            XT      RMB    2
                        END    START

NO ERROR(S) DETECTED
```


STOP PRESS! Friday 22nd September 6 p.m. – 7 p.m. Meeting of people interested in coordinating BUS STANDARDS. Venue: Conference Hall of **PCW** Show. Enquiries to *Bob Cottis* c/o PCW.



STATPACK

Colin Chatfield, MICRO-AID

In order to use large programs in a small amount of 'core' it is necessary to use some form of subtlety in coaxing the computer into acting like a giant. There are several ways this can be done and we have used the modular one in presenting a series that make up a statistical package.

In future issues we will build up a suite of programs that will give even a modest system the power of a machine several times its own size. Starting with the basic program for data entry and verification and going on to an information program and programs to do all the main statistical analysis that the majority of people will require.

The system is based on keeping the program in 'core' as small as possible in order that the data to be analysed can have as much memory as possible, thereby enabling more data to be processed. The data is entered into a matrix and stored on disc when not in use. Each sub-program can call this data from the file and process it separately. We have used the CHAIN command which brings in a program from another to do this. This is hidden in the MSIDOS V 1.2 interpreter and although not listed in the documentation it works very well. A disc system is necessary using this command and the data must be stored on the disc otherwise using CHAIN will destroy it. If you have CALL, as in MSIDOS V3, you need not store the data to be processed as it is not destroyed when calling any different package. In the latter case certain lines may be omitted from each program.

It is not the intention of this series to go into the technicalities of statistics or how to interpret the results but leaves that to the user. Some of the analysis to be covered will be ten different averages, chi-square and associated statistics, correlations, regressions, graphs, charts and relationships between answers to questionnaires. More advanced programmers will be able to add to the package if they wish to.

Stat 1 Main Statpack Program

Two files are used, STATFL2 which holds the number of data items, and STATFL1 which holds the data up to 255 x 255 items. These files must be created before running any part of the package, in the following manner:

CREATE STATFL1, REC=6, FILE=1000 (THIS LAST FIGURE CAN BE AS LARGE AS YOU LIKE UP TO 65025); CREATE STATFL2, REC=12, FILE=1 STAT1 takes up about 3K of memory; so using a SWTPC 6800 system and a 12K disc BASIC you would be able to process about 1100 items of data.

MSI BASIC uses a ? for a print statement and I have in future parts used ?# (Z9) for printout. This statement is set for whichever port you are using by a statement at line 20, Z1=(whichever port you intend the answers should be printed to). LINE=132 sets the carriage or screen width. Sub-routines are used extensively and LINE 9380 is used to clear and set the screen of a CT64 system. Lines 120 to 150 check if you require help, if you do STAT2 is chained for information. (See next article). Lines 1010 to 1135 set up the array size if you are not using stored data. Lines 1140 to 1240 are requests for the type of statistic you wish to do. Lines 9100 to 9140 are the data input. Lines 9450 to 9515 check the data and allow for correction. Lines 9530 to 9540 allow for corrected data to be entered before storing on disc. Lines 9600 to 9690 allow you to look at the data. Lines 9700 to 9790 save the data onto the disc. Line 9800 declares the arrays.

```
0005 REM - STAT1 MAIN STATPACK PROGRAM FOR ENTERING DATA
0080 LINE= 132: GOSUB9380
0100 ? TAB(22);"STATISTICS PROGRAM":?
0120 ? :INPUT"      DO YOU REQUIRE HELP - Y OR N ",A$
0130 ? : IFA$="N"THEN1010
0140 IF A$<>"Y"THEN120
0150 CHAIN STAT2
1010 INPUT "      NEW OR SAVED DATA ",A$
1015 IF LEFT$(A$,1)="S"THEN1135
1020 ? :?"ENTER DATA AFTER STATING # ROWS & COLUMNS;"
1022 ? " MAXIMUM OF 255 IN ONE DIMENSION"
1030 INPUT "      ENTER NOW ROW, COLUMN ",A,B
1032 IF A<2THEN?"AT LEAST 2 ROWS": GOTO1030
1034 IF B<1THEN1030
1036 IF A>255THEN1022
1038 IF B>255THEN1022
1040 IF A*B>1000THEN?"ARRAY TOO BIG, RE-ENTER": GOTO1020
1050 ? "YOUR ARRAY IS ";A;"X ";B;:?CHR$(8);:?". ";A*B;"ITEMS."
1070 ? :?"ENTER DATA ROW BY ROW. RETURN AFTER EACH ITEM."
1080 GOSUB 9800
1100 GOSUB 9000: GOSUB9450: GOSUB9700
1135 INPUT "DO YOU WISH TO SEE STORED DATA",A$:IFA$="Y"THENGOSUB9600
1140 ? :INPUT"WHAT STATISTICS DO YOU WISH TO DO? `SOS' FOR HELP ",A$
1160 IF LEFT$(A$,3)="SOS"THENCHAINSTAT2
1170 A3=ASC(A$):A3=A3-48: IFA3=0THEN1240
1180 IF A3=1THEN?"3=BASIC, 4=MOVING, 5=PROGRESSIVE ": GOTO1190
1185 GOTO 1200
1190 INPUT "      ENTER TYPE OF AVERAGE REQUIRED ",A1
1192 IF A1>5THEN1180
1194 IF A1<1THEN1190
1196 IF A1=3THENCHAINSTAT3
1198 IF A1=4THENCHAINSTAT4
1199 IF A1=5THENCHAINSTAT5
1200 IF A3=6THENCHAINSTAT6
1201 IF A3=7THENCHAINSTAT7
1202 IF A3=8THENCHAINSTAT8
1203 IF A3=9THENCHAINSTAT9
1204 IF A3=10THENCHAINSTAT10
1210 GOTO 1140
1240 ? TAB(20);"STATPACK END": END
9000 REM - SUB ROUTINES **********
9100 FOR I=1TOA:FORJ=1TOB
9120 ? "#";J;"   ";:INPUTC(I,J)
9130 NEXT J: IF I+1>A THEN9140
9135 ? "ROW # ";I+1;" NOW BEGINS"
9140 NEXT I: RETURN
9380 ? CHR$(25);:?CHR$(25);:?CHR$(22);:?CHR$(12);: RETURN
9450 GOSUB 9380:FORI=1TOA:B3(I)=0:NEXTI
9455 ? "CHECK DATA ROW BY ROW & RETURN IF OK. ";
9460 ? "IF DATA ROW IS INCORRECT THEN ENTER `X' ."
9465 ? :FORI=1TOA:FORJ=1TOB
9470 ? C(I,J);
9475 NEXT J:?:?"ROW ";I;:INPUTA$:IFA$<>"X"THEN9485
9480 B3(I)=I
9485 A4=A4+B3(I):NEXTI
9490 IF A4>0THEN?"INCORRECT ROWS ARE ";
9495 A4=0:FORI=1TOA:IFB3(I)<>0THEN?B3(I);
9500 A4=A4+B3(I)
9505 NEXT I:?:IFA4=0THENRETURN
9510 FOR I=1TOA:IFB3(I)>0THENGOSUB9530
9515 NEXT I:RETURN
9530 ? "INPUT NEW ROW DATA FOR ROW ";B3(I)
9535 FOR J=1TOB:INPUT"      ",C(I,J):NEXTJ
9540 ? :FORJ=1TOB:?C(I,J);:NEXTJ:?
9550 INPUT "      IS DATA OK ",A$
9560 IF A$="Y"THENRETURN
9570 GOTO 9530
9600 IF A7=1 THEN9670
9605 OPEN #10,STATFL1 FOR INPUT
9610 OPEN #20,STATFL2 FOR INPUT
9620 FIELD #10,F=6
9630 FIELD #20,A=6,B=6
9640 SET #10=1:SET#20=1:GET#20
9650 GOSUB 9800
9660 FOR I=1TOA: FORJ=1TOB: GET#10: C(I,J)=F: NEXTJ: NEXTI
9670 ? : FORI=1TOA: FORJ=1TOB: ?C(I,J);: NEXTJ: ?: NEXTI: ?
9680 IF A7=1 THENRETURN
9690 CLOSE #10:CLOSE#20:RETURN
9700 OPEN #10, STATFL1 FOR UPDATE
9710 OPEN #20,STATFL2 FOR UPDATE
9720 FIELD #10,F=6
9730 FIELD #20,A=6,B=6
9740 A4=A:A5=B:SET#10=1:SET#20=1:GET#20
9750 A=A4:B=A5:REWRITE#20
9760 FOR I=1TOA:FORJ=1TOB
9770 GET #10:F=C(I,J):REWRITE#10
9780 NEXT J:NEXTI
9790 CLOSE #10:CLOSE#20:RETURN
9800 DIM C(A,B),B3(A): A7=1: RETURN

THE GHOST
#
```

THE HARD KEYBOARD

W. McIvor

A Byte Orientated Hex Keyboard

When designing my MPU system I decided that input and output should be completely software independent. This would allow the use of a relatively small monitor program in 256 bytes of CMOS memory with battery backup in place of the usual ROM monitor. The use of ROM was avoided because it would involve either expensive programming charges during the development of the monitor program or the building of a programmer and UV-eraser. CMOS RAM on the other hand is relatively expensive and therefore the monitor program has to be as compact as possible without sacrificing any of the monitor's power. This is achieved by using input and output hardware which requires minimal software to communicate with the MPU.

Software Intensive Keyboards

Most commercial hex keyboard use a software scanning technique which scans a matrix of *push to make* switches one column at a time until a closed switch is found. The *row* on which this switch is located is *read into* the MPU and used *with* the column information to determine the appropriate hex digit. This method has two disadvantages. Firstly, a lot of software is required, not only to find the closed switch but also to convert the information into a hex digit. Secondly, the keyboard is only active when the program requires it and therefore cannot be used to interrupt a program.

Where a hardware hex keyboard is used it generates a four bit binary number and a *data-ready* signal whenever a key is depressed. This simplifies software substantially but software still has to allow for keyboard switch contact "bounce" and has to pack the four bit numbers into eight bit bytes.

Self Debouncing

A hex keyboard was therefore required which would *debounce itself,* and assemble a full byte of data before setting the data ready flag. This reduces the software requirement to testing the data-ready flag and loading data from the keyboard if the flag is set, the loading of data automatically resetting the data ready flag.

Fig 1 shows a block diagram of the circuit. With no key pressed the strobe and hex data lines are held at logic '1' by the 1K pull-up resistors. When a key is pressed the strobe line and the appropriate data lines are taken to logic 'O' via diodes.

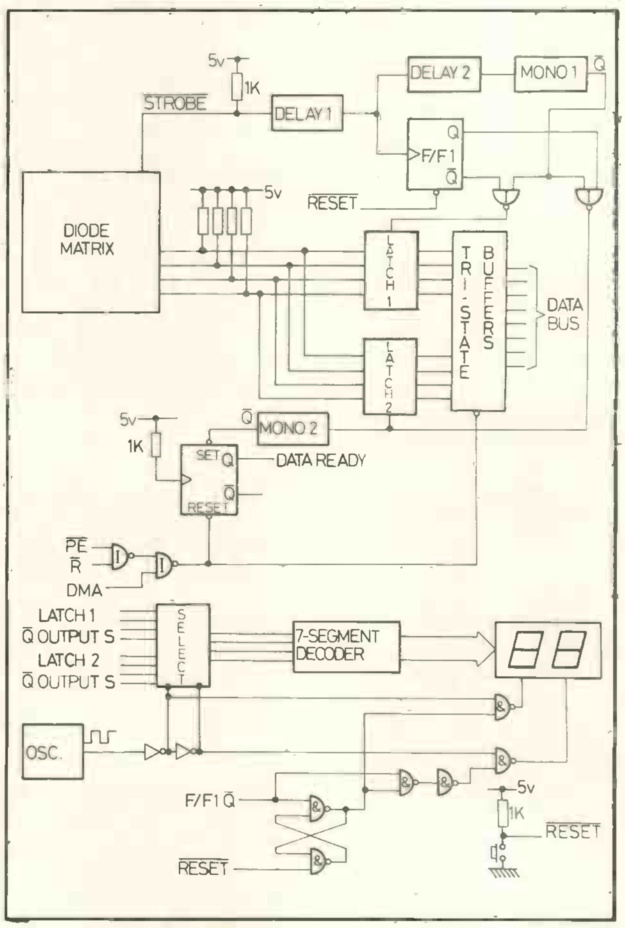


Figure 1

The first delay takes care of any contact bounce and then clocks the first flip flop (F/F 1). The second delay is to ensure that the flip flop has time to change over before the latch clock pulse is sent from mono 1. Initially F/F 1 is cleared (Q = 'O'). Therefore, when the first hex digit is keyed in $\overline{Q}$ becomes logic 'O' followed by a negative pulse from mono 1 which clocks the data into latch 1. When the second digit is keyed in Q becomes logic 'O' and the pulse from mono 1 clocks latch 2. The rising edge of this clock pulse triggers mono 2 which sets the second flip flop (F/F 2). The Q output of F/F 2 is therefore the data ready signal. A 'read' performed by the MPU will clear F/F 2 and enable the tristate buffer which connects the outputs of the latches to the data bus allowing the eight bit byte to be read.

The second part of the circuit is used to display the data entered. This again saves software and leaves the main output free for the rest of the program. A square-wave oscillator alternatively selects the output from each latch as the input to a binary to 7-segment hex decoder. At the same time, the appropriate digit drive buffer is also enabled. After keying in the first digit the second digit must be blanked. This is achieved by using the Q output of F/F 1, which is logic 'O' after the first digit is keyed in, to disable the drive buffer for the second digit.

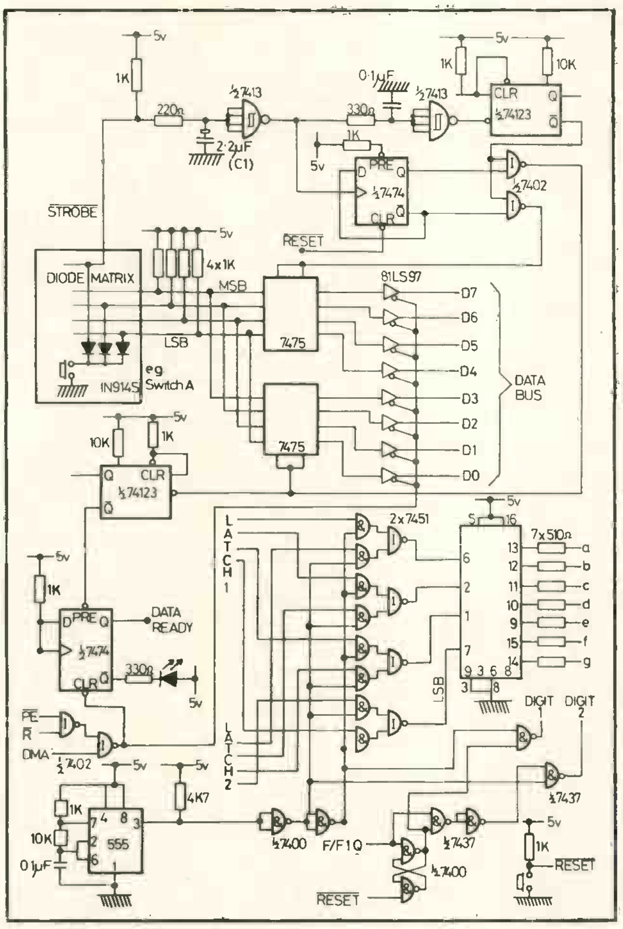


Figure 2

Reset System

Finally, a reset system is required to ensure that when the power is first turned on the first keyed digit goes into latch 1. The reset button clears F/F 1 and sets an R-S flip flop constructed from two nand gates. This flip flop blanks both displays. $\overline{Q}$ from F/F 1 is used to reset the R-S flip flop after the first digit is keyed in allowing normal operation to continue. The reset button can also be used to erase an incorrect first digit but not to correct the second digit.

Fig. 2 shows the complete circuit diagram. The delays are produced using 7413 schmitt triggers. For example the strobe line is initially high and C1 is fully charged. When a key is pressed the strobe line goes low and C1 discharges to o.7V, the voltage drop across the diode. The output of the schmitt trigger goes high when C1 has discharged to it's lower trigger voltage. The time taken for C1 to fall to this voltage is the delay time. Contact bounce is eliminated as long as the delay time is longer than the contact bounce time.

Conclusion

Several points should be noted. The data selection for the display is performed using 7451s which invert the data. The inputs are therefore derived from the $\overline{Q}$ outputs of the latches.

Because the monostables only need to produce very short pulses no timing capacitors are required. The timing resistors must however be included.

The diode matrix for the keyboard gives a logic 'O' of 0.7V whereas TTL logic 'O' is defined as less than 0.8V. This means noise immunity is only 100mV. In the original circuit, with a 0.1μF decoupling for every three ICs this caused no problems, but germanium diodes could be used if felt necessary.

The original circuit was designed for use with the SC/MP which uses memory mapped peripherals. Data ready was connected to one of the sense inputs. $\overline{PE}$ was derived from an address decoder and $\overline{R}$ is the MPU read strobe.

When the input marked DMA is taken high the keyboard is connected to the data bus allowing it to be used as a data input device when using direct memory access to write data into memory.

PCW OPEN PAGE

The Amateur Computer Club View

Mike Lord

Homer G. Homebrew

ON STANDARDS, AND OUR PLACE IN THE SCHEME OF THINGS

A computer is a complicated beast. One which can be described on many different levels, and must be, since the only way we can understand it is by abstraction. Take for example the BASIC statement;

LET A = B + C

Fairly obvious what this means. Now think about current flowing in a wire. Easy to visualise, and any A level physics student could describe it in terms of Field Theory. Now describe the execution of the BASIC statement in terms of Field Theory.

Impossible. But, one can use Field Theory to describe the behaviour of conductors & semiconductors, and so gain an understanding of how a transistor works. From this one can abstract the idea of a transistor and its principal characteristics, and then use the concept 'transistor' to understand how a NAND gate works. The abstract idea of a NAND gate gives us an understanding of registers, memories, and the other circuit blocks used to build part of a computer. And having an abstract (or 'programming model' or 'block diagram', call it what you will) of the computer hardware, one can then move up the levels of software; machine language, assembler, high level language etc. At any one level one can understand how the machine works but you can't describe the minute details of its operation, or describe why it is doing what it is without changing the level of abstraction.

Because the computer is a complicated beast it takes many man-hours to design the hardware, and many more to produce the software. At a professional, full time, level it takes at least three man months to design a typical board, and six man months to produce a decent BASIC interpreter. For the average amateur, working with limited resources one or two evenings a week, to design a half-way decent computer from scratch, with a reasonable amount of software, would take years.

So the amateur, as well as the professional, has to choose the level in which he will operate. Whether he is to design logic gates or COBOL compilers, he has to work with an abstraction of what the people working at a 'lower' level produce ('transistors' or 'assembler code'), and he has to accept whatever he is given as sacrosanct, unalterable; in other words, a STANDARD.

STANDARDS needn't be decided by committee. In fact committees usually just accept what has been thrown up by the marketplace, changing only the names to protect their reputations. 74 TTL, the S100 bus, and FORTRAN weren't designed by committees, but surely they are real STANDARDS in the computing world.

What a committee can do, however, is *crystalise* a new STANDARD. If the time is right, then a review of new ideas, coupled with a determination to fully support their final decision, can bring forward a new STANDARD to be used and cursed for many a year.

Therefore, the ACC would like to re-open the question of cassette tape standards for amateur use. In 1975, a conference at Kansas City adopted the CUTS (Cassette Users Tape Standard) from a choice of about a dozen systems. It has become widely adopted and is reasonably satisfactory for the exchange of data between different hobbyists; however, it is agonisingly slow (300 baud) and there are a whole host of faster post 1975 developments now available for comparison.

Anyone having views on this subject is invited to draw up a brief outline of his position, preferably as a series of short numbered notes, and send them together with any proposed circuit or other details to Alan Secker, 209 Albury Drive, Pinner, Middx. HA5 3RH. All proposed systems will be compared and contributors with like views will be put in touch with one another with the intention that a debate will be staged with the minimum of duplication.

Alan Secker would also like to hear of any other areas which amateurs consider it would be worthwhile considering for possible standardisation exercises.

NEW ENCOUNTERS

Anyone interested in forming an amateur computing club in the Leicester area is invited to contact Mr. G.B. Foden, 11 Gaddesby Lane, Rearsby, Leicester. (Telephone, Rearsby 247).

A Computer User Group has been formed at the College of Higher Education, High Wycombe, by Roy Woodbridge and Ken Spencer of the Department of Engineering.

Colchester enthusiasts should contact Donald Clarke, 21 The Avenue, Colchester CO3 3PA (Tel: Colchester 66637).

Jim Turner, 63 Millais Rd., London E11 4HB, has offered to co-ordinate the resurgance of the London Group, and invites anyone willing to help to send him an s.a.e.

Mr. N. Beard, of High St., Braithwell, Rotherham, S. Yorks wants to hear from anyone else in the area with an interest in computing. (The address is correct, no street number!)

Eltham enthusiasts are invited to get in touch with Mr. R. I. Mitchell, 58 Kenilworth Gardens, Shooters Hill, London SE18 3JB (Tel: 01-856 2489) as he would like to start a local group in that area.

Finally, Norman Fox and Tom Turnbull feel that an independent PET User Group (PUG?) would be worthwhile, and ask anyone interested in joining to contact Norman at 22 Firs Walk, Tewin Wood, Welwyn, Herts (telephone: Bulls Green 433).

The SUACC have just formed a sister club called Southampton Amateur Computer Coub. This club is open to anyone, whether connected with Southampton University or not, and we ambitiously hope that this will be the start of a major nucleus for personal computing in the South. In this connection we are very willing to contact individuals who would like the assistance and pool of knowledge that the club can provide.

Paul G. Dorey,
Advisory Editor,
Benchmark,
(SUACC Newsletter)

MICROPROCESSORS

*A ten week course on Wednesday evenings starting October 11th, 1978.

Outline Syllabus.

Internal organisation, bus structure, clocking, timing diagrams, control signals, input/output, interfacing, interrupts, machine code, addressing modes, memory organisation, ROM and RAM, monitor programs . . .

The syllabus is designed to provide a conceptual framework within which a microprocessor system can be understood in terms of the co-operative relationships between a number of relatively simple devices: MPU, memory, buffers, I/O controllers etc.

People with some knowledge of software *or* hardware will be welcome on the course, which attempts to bridge the gap between the two.

To illustrate the material taught we have a 6800 system, a Z80 system, and an idealised low level language which is simulated on the college mini computer.

Approximate cost £25.

Willesden College of Technology,
Denzil Road, London NW10 2XD
Tel: 01-459 0147/8/9

THE NEWBEAR COMPUTING STORE

Hardware Components Section

Goods are normally shipped within 24 hours subject to availability.
Barclay card & Access VAT at 8% for Hardware Components.
30p postage and packing unless otherwise stated.
Cheques to be made out to 'The Newbear Computing Store'

Send for an up-to-date catalogue to:

The Newbear Computing Store
7 Bone Lane, Newbury.

Callers welcome Monday to Saturday 9.00a.m.-5.30p.m.
The Newbear Computing Store is a division of Newbury Laboratories Ltd.

VIM 1

The new 6502 based micro from Synertek.

Fully assembled and tested	£199.00
Send for data sheet	carriage £1.00

S-100 bus

Z-2 kit (CPU card + frame)	£395.00
BYTESAVER PROGRAMMING BOARD KIT (2708)	£ 95.00
TU-ART Interface Kit	£130.00
16K byte Dynabyte ram card (fully tested)	£415.00
North Star Mini Floppy Kit	£490.00

BEAR BAGS

1.	77-68 CPU P.C.B. Components and edge connectors	£ 49.50
2.	77-68 Leds and Switches	£ 14.95
3.	77-68 Power supply*	£ 17.95
4.	77-68 19" 5 u Rack and Backplane*	£ 27.70
5.	77-68 4K RAM P.C.B. and components	£ 75.00
6.	77-68 MON 1 P.C.B. and components	£ 50.70
7.	4K Ram Exorciser P.C.B. and components	£ 71.50
8.	8K Ram Exorciser P.C.B. and components*	£160.00
9.	Petitevid V.D.U. Kit	£ 85.00
10.	Kansas City Casette Interface	£ 18.95
11.	U.H.F. Modulator	£ 4.50
12.	77-68 V.D.U. P.C.B. and components	£ 69.50

Plus VAT 8% 50p p&p *£1.00 Postage & Packing

Zilog Microcomputing I.C's

Z80 CPU 2.5 MHz	£15.50
Z80 P10 PS	£10.00
Z80 CTC PC	£10.00
Z80A CPU* 4 MHz	£20.50
Z80A P10*	£11.55
Z80A CTC	£11.55

Motorola Microcomputing I.C's

MC 6800	£11.25
MC 6820	£ 6.20
MC 6850	£ 6.74
MC 6810AP	£ 3.61
MC 8602P	£ 2.88
MC 14536P	£ 3.69
MC 3459	£ 2.53

MEMORIES

2102–1	£ 1.25
2102L–1	£ 1.35
2112	£ 3.04
2513	£ 6.50
SWATBUG	£16.00
4027	£ 4.50
2114	£10.00
4116	£19.00
2708	£10.50
MC6830L7 (MIKBUG)	£13.65
2716	£22.50

Microprocessors

SC/MP 11 (INS 8060N[ISP8A/600N]	£10.30
6502	£14.93
F-8 and 8080 parts available soon.	
INS 8154N	£ 8.18

BUFFERS

81LS95	£1.30
81LS96	£1.30
81LS97	£1.30
81LS98	£1.30
8T26	£1.84
8T95	£1.60
8T97	£1.60
74367	£1.30

INTERFACING I.C's

SFF 96364	£16.20
MC 1488p	£ 1.40
MC 1489p	£ 1.40
75150p	£ 1.30
75150n	£ 1.20
75154	£ 2.30
4N33	£ 1.95
AY-5-1013	£ 5.30
6402	£ 4.50

Petitevid VDU Kit £85.00

- 110 to 1200 based V24
- 64 ch. X 16 lines Scrolling
- All on 8" x 4" P.C.B.

Needs TV set. VHF modulator and ASCII keyboard.

77-68 P.C.B's

77-68 CPU	£10.00
77-68 4K Ram	£10.00
77-68 Mon 1	£10.00
77-68 VDU	£10.00
Licon ASCII Keyboard Fully Assembled + tested	£86.00 (£1.50 p+p)

BOOKS

All books are subject to availability.

		P. & P.
COMPUTER DESIGNS		
77-68 6800 Microcomputer	£ 7.50	.50
Spare diagram set for 77-68	£ 1.50	.50
WB–1 TTL Microcomputer	£ 6.50	.50
Spare diagram set for WB–1	£ 1.00	.50
FROM ADAM OSBORNE ASSOCIATES		
Introduction To Microcomputers		
Volume 0: The Beginners Book	£ 5.95	.50
Volume 1: Basic Concepts	£ 5.95	.50
Volume 2: Some Real Products June 1977 Revision	£11.95	1.00
8080A/8085 Assembly Language Programming	£ 6.95	.50
Some Common Basic Programs	£ 5.95	.50
6800 Programming for Logic Design	£ 5.95	.50
8080 Programming for Logic Design	£ 5.95	.50
Payroll with Cost Accounting in Basic	£ 9.95	1.00
SCELBI		
Understanding Microcomputers & Small Computer Systems	£ 7.56	.50
Scelbi 6800 Software Gourmet Guide & Cook book	£ 7.95	.50
Scelbi 8080 Software Gourmet Guide & Cook book	£ 7.95	.50
8080 Standard assembler	£15.95	.75
The Scelbi Byte Primer (replaces Best of Byte)	£ 9.95	1.00
The 8080 Programmers Pocket Guide	£ 2.35	.30
ZILOG		
Z-80 Technical Manual	£ 4.00	.50
Z-80 P10 Technical Manual	£ 2.50	.50
Z-80 Programming Manual	£ 4.00	.75
MOTOROLA		
Understanding Microprocessors	£ 2.75	.30
M6800 Microprocessor Programming Manual	£ 4.50	.50
M6800 Microprocessor Applications Manual	£ 9.50	1.00
SYBEX		
The Home Computer Revolution	£ 3.45	.50
Computer Lib	£ 5.95	.50
Memory Databook by National Semiconductor	£ 2.50	.75
TTL Databook by National Semiconductor	£ 2.50	.75
Algol for Beginners by D. Leach & R.T. McCartney	£ 2.00	.50
Games, tricks & Puzzles for a hand calculator by W. Judd	£ 2.49	.50
Microcomputers KIM-1 user Manual	£ 5.00	.75
73 Amateur Radio by Longis Tufts	£ 2.00	.30

6500 Hardware Manual	£ 5.00	.75
6500 Programming Manual	£ 5.00	.75
The best of Creative Computing Volume 1.	£ 6.95	1.00
The best of Creative Computing Volume 2.	£ 6.95	1.00
PCC's Reference book of Personal & Home Computing	£ 4.95	.50
Texas TTL DATABOOK	£ 7.00	.75
Data Acquisition Handbook	T.B.A.	.50
The Z-80 Microcomputer Handbook	£ 7.25	.50
SC/MP Microprocessor Applications Handbook	£ 3.00	.50
Microprocessor Manual 2650	£17.00	.75
Microprocessor 2650	T.B.A.	.50
Getting Involved with your own Computer (A Guide for Beginners)	£ 4.75	.50
BASIC		
Instant Freeze-Dried Computer Programming In Basic By Jerald R. Brown	£ 4.95	.75
My Computer likes me when I speak in Basic	£ 1.65	.30
Computer programs that work	£ 2.40	.75
Basic Software Library Volume 1:	£17.50	.50
Volume 2:	£17.50	.50
Volume 3:	£26.95	.50
Volume 4:	£ 7.95	.50
Volume 5:	£ 7.95	.50
Volume 6:	T.B.A.	.50
Volume 7:	£26.95	.50
What to do after you hit Return	£ 7.00	.75
DILITHIUM		
Home Computers by Dilithium:–		
Volume 1. Hardware	£ 5.95	.50
Volume 2. Software	£ 5.95	.50
Home Computers: A beginners Glossary and Guide	£ 4.95	.50
Beginning BASIC	£ 4.95	.50
A Step By Step Introduction to 8080 Microprocessor Systems	£ 5.70	.50
Take A Chance With Your Calculator	£ 4.95	.50

24 HOUR TURNROUND ON ORDERS

**Securicor delivery advised please phone for charges.*

Please add 8% VAT to all prices. P & P 30p unless otherwise stated. Barclaycard & Access welcome. Overseas orders issued with Pro-Forma Invoice. Send for catalogue to **NewBear Computing Store, Bone Lane, Newbury, Berks.** Callers welcome Mon – Sat, 9.00 – 5.30 but please phone us first on 0635 49223. New office – 2 Gatley Road, Cheadle, Cheshire (callers only) Tel: 061 491 0134

The GREAT PERSONAL COMPUTER WORLD SHOW

General

The Exhibition and Conference aim to show the latest developments in the field of micro-processor technology and how they can be applied to the small business, in education and research, and by the computer hobbyist. The Exhibition will highlight applications and benefits of micro-processors which increase business efficiency, save time and money.

21 – 23 September
Three Days to Remember
PCW and *NCC* **get together ....** *International Chess Master David Levy to co-ordinate Microchess Championship*

The Exhibition

The equipment on display will cover all aspects of hard and software, components and systems up to a price of about £20,000. Over 40 manufacturers will be showing their latest products – the list of exhibitors already includes – Crofton Electronics, Tandy Corporation, Micronics, Bywood Electronics, Research Machines, Mutek, Comart, Nascom, Personal Computers, Computer Workshop, Star Devices, Belvedere Computer Services, Cytek UK, Technologics, Strumech Engineering, Datac, Sirton, Pelco, Collins Consultants, Newbear Computing Store, Sintrom ..... Stands will have been fully booked by mid-August.

The Features Area

Among the main attractions to be included in the area will be the voice-controlled Sol Computer presented by Leslie Solomon, the 'father' of personal computers in the U.S.A.; a somewhat upgraded version of CAP Microsoft's computer controlled model railway previously exhibited at the IEA Show in Birmingham; and various personal exhibits by private individuals.

Would *you* like to show off your homebrew system or your innovation? Write to *PCW* immediately. Mark your envelope "Homebrew" or "Innovators' Corner".

The PCW Competitions

Enter the Competition Now. Write to **PCW** immediately for the rules of the Competitions.

- Best Software — £200.00 prize
- Best Homebrew System — £200.00 prize
- Best School Application — £200.00 prize
- Best Home Application — £200.00 prize
- Chess — £200.00 prize

• **The PCW Microchess Championship.** The international chess master, **David Levy,** is the **PCW** advisor on all aspects of the Microchess Championship, and has drawn up its rules.

David Levy is a leading world authority on the subject of Computer Chess. In 1968 he started a bet, now worth £1,250, that no computer program would beat him in a match within ten years. That bet is due to expire at the end of August when Mr. Levy expects to collect his winnings. Since 1971 he has been the tournament director and commentator at every major computer chess tournament, including seven North American and two World Championship events.

Write now to **Competitions, PCW, 62a Westbourne Grove, London W2.**

The National Computing Centre Enters the Show

The National Computing Centre has generously donated £200 prize money for the 'best home applications' category which includes software for home accounting, timetables, central heating control and recipes. Brian Stanford-Smith of the National Computing Centre has advised **PCW** on the drawing up of the rules of the competitions, and will be one of the panel of judges.

Prizes for runners-up from manufacturers will also be awarded and so far Newbear Computing has donated a Petitevid terminal, and Lynx Electronics has donated a Nascom 1 hobbyist kit.

The Conference

The Conference programme is tailored to a particular interest each day. The timetable allows delegates to follow-up points individually with speakers at coffee, lunch and tea breaks in addition to the set discussion periods. In addition to the speakers listed below, Leslie

Solomon, Technical Editor of *Popular Electronics* in the United States and regarded as the father of home computing across the Atlantic, will be available at discussion sessions.

The Conference programme will start promptly at 09.30 each day and finish at 17.00. Delegates should check in no later than 09.15.

The Conference will give the businessman, educationalist and hobbyist the chance of updating their knowledge on the applications of microprocessors, of learning more about them and of discussing them with the experts.

Thursday 21 September Micro-Processors for the Small Business
Starting with first principles, expert advice on all aspects of applying micro-processor techniques for greater efficiency in the small business.

Mike Gurr BSc (Eng)
Data Base Consultant BOC Ltd
The Language of Business Computing (Basics for the Business Beginner)

John Burnett
Computer Workshop
Case History of a Beginner in Business Computing
(How to Accelerate the Learning Process)

Laurence Payne
Accountant and Principal Computech Systems
Micro-Computer Applications Economics v Technology — Which is more important?
(General review including hardware and software developments and availability)

Richard Waller FCA BCS
Consultant CAP Microsoft
Acquiring the Software for a Small Business

David Hebditch FBCS MIDPM
Consultant
How to Choose a Small Business System
(Practical Guidelines for Evaluation and Selection)

Friday 22 September Micro-Processors in Schools and Universities
The teaching and application of micro-processor techniques in schools and universities; fascinating case histories (and demonstrations) of the use of these techniques in medical research and to help the handicapped.

Derek Esterson MSc
ILEA Inspector for Computer Education
Development of Computer Education in London Schools
(A General Survey from Mainframe to Micro)

Colin Wells MSC BSc
Master in Charge, Computing, Down School, Dartford
Use of Micro-Computers in a Secondary Modern School
(Demonstration of Programmes for use with children including a 5-a-side football game)

Professor G D Dawson MSc MB ChB
Professor of Physiology, University College London
Micro-Computers in the Teaching of Laboratory Research Methods
(Application to Investigation of the Nervous System. A Demonstration of Some Capabilities of Minimum Systems)

Frank Lovis MBCS
Senior Lecturer in Mathematics at the Open University
Student Computing at the Open University — Including Future Plans

Julia Howlett BSc
Project leader of MAVIS, General Consultant on Micro-Processor Information Systems
Some Special Aspects of Micro-Processor Driven Personal Information Systems

Saturday 23 September Micro-Processors for the Hobbyist and Beginner
An opportunity for delegates to hear from the experts what is possible and how to achieve it.

Mike Dennis
Consultant to Personal Computer World
An Overview of Computer Architecture — A Review of First Principles of Micro-Processors

John Miller-Kirkpatrick
Bywood Electronics
Soft/Hardware — How to Treat Your Micro-Processor as a TTL/CMOS Simulator Including the Application of PORTs etc

David Goadby
Computer Consultant
The Gentle Art of Interfacing — How to Make Your Processor Communicate

Sheridan Williams LIMA AFBIPS
Lecturer in Computer Science Barnet College of Further Education
An Introduction to Computer Programming
(Including a Simple Explanation of the Rules of BASIC)

John Coll
Department of Electronics Oundle School
Assemblers and Editors
(An Introduction to Machine Language Processing)

Conference Fees
Charges include coffee, tea, VAT and printed speakers' abstracts. The Conference admission pass entitles delegates to free entry to the Exhibition on any day.

Thursday and Friday (including finger buffet lunch)		£20.00
Saturday	Whole Day	£9.00
	Half Day	£5.00

Please apply for Conference tickets on the form by 8 September.

The Show Catalogue
The centre section of the next issue will feature the Show's catalogue containing a full run-down on each exhibitor together with a plan of the Exhibition and detailed information on the Conferences.

Show Time:
21 — 23 September. Three days to remember.

The Show is open: 10.00 — 19.00 Thursday 21 Sept.
10.00 — 19.00 Friday 22 Sept.
10.00 — 17.00 Saturday 23 Sept.

The Venue
The West Centre Hotel is situated in Lillie Road, London SW6. The closest London Transport Underground Station is West Brompton (5 minutes walk); and buses 30, 74, 74b pass the Hotel.

How to get your tickets
Book now by filling in the application form below and sending it together with the appropriate remittance to **Will Martin, Interbuild Exhibitions Ltd.,** 11 Manchester Square, London W1M 5AB. Telephone: 01-486 1951

Personal Computer World Show 21-23 September 1978
West Centre Hotel London
Application for Tickets

Conference

		Cost	No. of Tickets
Thursday 21 September	Small Business Day	£20*	
Friday 22 September	Schools and Universities Day	£20*	
Saturday 23 September	Beginners and Hobbyists Day	£9* Whole Day £5* Half Day	

**Includes free admission to exhibition.*

Exhibition

Please send me . . . . . . tickets @ £1.00 each. I do not wish to attend the conferences.

I enclose a cheque for £ . . . . made payable to Interbuild Exhibitions Ltd.

Name . . . .

Company/Organisation. . . . .

Address . . . .

. . . . Telephone No. . . . . .

A Tidal Wave

Boris Sedacca

Leslie Solomon Technical Director of *Popular Electronics* magazine, with a message for our readers, conveyed through **Boris Sedacca.**

"My main purpose will be to make people in the UK aware of the State-of-the-Art in the USA, and where we stand today, because I believe that this is where the UK will be in a year's time.

All this may sound very well, but the most important aspect to be considered is how the computer can be used to work for people. There is a certain mystique which surrounds the expression 'computers' and I wish to see people regarding them as no more than tools to be used.

For this reason I want to demonstrate some of the many possible applications other than those more conventional applications commonly associated with computers such as science, business and ruling the world.

Everything I will be demonstrating will be aimed at non-computer people. For example if you have a television, you use it. Do you know how it works? Do you care? No, you just turn it on.

The word 'computer' scares most people. This is fuelled by the influence of the cinema, television and other media. People associate computers with flashing lights, huge reels of magnetic tape, and white dust coats. I want to show that it is no more than an utilitarian tool.

In the US, I am the chairman of the Committee for the Handicapped. We do not regard the home computer merely as a game playing machine — we see it as a means for the handicapped to set up communication with the universe.

We are looking at ways of aiding people who are, for instance, totally paralysed. Some of them cannot even talk let alone move. Whatever faculties they do possess we try to harness them. For example, if they are capable of uttering any sounds, these may be input through a voice synthesiser in a predetermined code. The computer may be trained to recognise a certain voice pattern and turn lights on and off, adjust temperature controls or warm up a meal in the oven.

There are numerous musical applications which I hope to demonstrate too. The relation between mathematics and music makes it an ideal application for the computer.

This is useful for music students. They do not need to know how a computer works in order to produce music on it, the same as they do not need to know how a piano or a synthesiser works in order to play it.

The flashing lights, switches and buttons must go too; this is one thing I realised when I was working on the Altair microcomputer. They just scare people away. This is why an increasing trend among personal computer manufacturers is to integrate the computer circuitry within a keyboard casing and nothing else. This in fact is how the Sol is constructed — just a keyboard on the face of it.

What people will have to realise is that there is no more intelligence in a computer than in a cash register. No computer ever made has solved a problem. Humans solve the problems. They are, and will always be superior. The only thing a computer can do by itself is get rusty.

My little girl plays around with a computer at home, and a friend of mine once asked her, 'aren't you afraid that the computer may one day take over and rule the world?' Her reply was 'no chance; I know where the power plug is'.

Now that the average man-in-the-street is involved in computers they should not be made to look complicated. If he gets into a car, he does not want lights and guages indicating temperature conditions, wind, tyre pressure, etc. He is not running a submarine.

Lights are at best confusing. When they come on does it mean it's good or bad? I believe in one light emitting display; power-on.

In the long run, humans will have to learn to control computers, just as they learned to control past technologies, or even in fact wild animals. The horse or the dog had to learn to adapt to man — not vice-versa. As far as I am concerned, it is an answer looking for questions.

In hardware terms what I will be exhibiting is a 'speechlab' facility in which I participated. Speechlab allows one to talk to a computer and for the computer to reply by way of voice synthesis, or to activate controllers attached to appliances, central heating, lights, etc.

I hope also to have on display in the features area a computer that can play music and to display the use of high quality colour graphics on a Sol computer.

At the show I will be participating in the discussion sessions in between the presentations. I will be there to help in any way I can to get the show on the road.

My message to you all in the UK is: there is a tidal wave coming over from across the ocean. There are a couple of sharks as well so you'll have to be careful. The water is lovely otherwise."

Bus of the Century

THE S100 BUS

Francis E. Cox
Technical Director, Interam

The release of the MITS Altair 8800 personal computer in the USA in January 1975 began the era of the home microcomputer. This machine, together with the IMSAI 8080, established the S100 ("Standard-100") bus as the leading microcomputer bus structure. The purpose of a computer bus is the support of high speed information transfer between the major components of the computer.

These are: the central processor, the memory and the input/output devices. Since the two original home computers used the Intel 8080 microprocessor as the central processor, the bus was designed around the 8080 signals. This has not however, as will be seen, limited its applicability to this processor; now over 150 manufacturers make boards compatible with the S100 bus system.

Technical aspects

The following section is a summary of the technical aspects of the S100 bus; it may be skipped if you are not really interested in hardware.

The signals on the S100 bus are functionally divided into five groups: a) the Address lines, b) the Data lines, c) the Status lines, d) the Command/Control lines and e) miscellaneous lines.

The 16 Address lines hold the address of the memory byte being read from or written into during a memory read or write cycle; up to 65, 536 8-bit bytes may be individually accessed. Additionally, during an I/O port read or write cycle the address lines hold the address of the port (0 to 255) on the first eight of the lines, with the same pattern repeated on the second eight. The Address lines are notated A0 to A15 (inclusive); the numerical suffix in all groups designates the position of the line in the group, with 0 indicating the least significant bit.

The 16 Data lines are split into the input lines to the CPU D10 to D17; and the output lines from the CPU D00 to D07. Both the input and output data paths are, clearly, eight bits wide.

The 8 Status lines and their functions are as follows: SINTA– acknowledge interrupt request; /SWO (/ indicates signal is active low) – current operation is a write to memory or output to port; SHLTA – acknowledge halt instruction; SOUT – address bus holds the address of selected output port; SMI – machine is in op-code fetch cycle; SINP – address bus holds address of selected input port; SMEMR – data input bus is to hold data from memory.

The 5 Command/Control lines and their functions are as follows: PHLDA – CPU acknowledges and enters hold mode (used for direct memory access – DMA); PSYNC – CPU is beginning a machine cycle; PDBIN – the input data bus strobe (lines D10-D17); /PWR – the output data bus strobe (lines D00 - D07); PWAIT – CPU acknowledges wait request.

There are a number of miscellaneous other signals on the S100 bus; the main ones are as follows: XRDY, PRDY – addressed device ready to accept data from or send data to, the CPU (wait state request); /PHOLD – hold (suspend) CPU request; /PRESET – reset the CPU; /ADDR DSBL, /D0 DSBL, /CC DSBL, /STATUS DSBL – disable the address, data output, command/control and status lines respectively; /PINT – interrupt CPU request.

Additionally, there are power supply inputs (unregulated) at +8V, +16V, –16V and a ground, of course.

Comprehensive details of the timing characteristics of the signals mentioned above can be found in the references listed at the end of the article.

Power and versatility

Although the S100 bus may appear at first sight to be rather complex, it is in fact very powerful and versatile. An example of this is the ease with which direct memory access (DMA) may be accomplished. Here a device such as a magnetic disk controller, plugged into the bus, takes control of the system over from the CPU and reads data into or out of memory at very high speeds. This avoids the bottleneck effect of having to pass all data via the CPU. This powerful concept of the system having different bus "masters" and bus "slaves" at different times can lead to very efficient data processing.

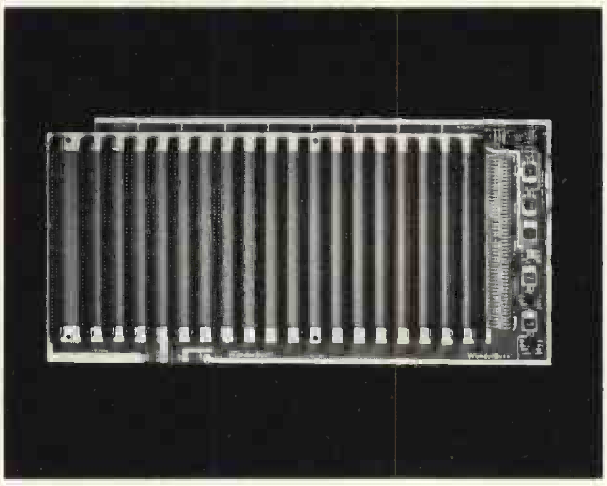
THE S-100 BUSS LAYOUT

pin 1	+8 Volts	pin 51	+8 Volts
pin 2	+16 Volts	pin 52	–16 Volts
pin 3	XRDY	pin 53	SSW DSB
pin 4	VI0	pin 54	EXT CLR
pin 5	VI1	pin 55	
pin 6	VI2	pin 56	
pin 7	VI3	pin 57	
pin 8	VI4	pin 58	
pin 9	VI5	pin 59	
pin 10	VI6	pin 60	
pin 11	VI7	pin 61	
pin 12		pin 62	
pin 13		pin 63	
pin 14		pin 64	
pin 15		pin 65	
pin 16		pin 66	
pin 17		pin 67	
pin 18	STATUS DSBL	pin 68	MWRITE
pin 19	C/C DSBL	pin 69	PS
pin 20	UNPROTECT	pin 70	PROTECT
pin 21	SS	pin 71	RUN
pin 22	ADDR DSBL	pin 72	PRDY
pin 23	DO DSBL	pin 73	PINT
pin 24	ϕ_2	pin 74	PHOLD
pin 25	ϕ_1	pin 75	PRESET
pin 26	PHLDA	pin 76	PSYNC
pin 27	PWAIT	pin 77	PWR
pin 28	PINTE	pin 78	PDBIN
pin 29	A5	pin 79	A0
pin 30	A4	pin 80	A1
pin 31	A3	pin 81	A2
pin 32	A15	pin 82	A6
pin 33	A12	pin 83	A7
pin 34	A9	pin 84	A8
pin 35	D01	pin 85	A13
pin 36	D00	pin 86	A14
pin 37	A10	pin 87	A11
pin 38	DO4	pin 88	DO2
pin 39	DO5	pin 89	DO3
pin 40	DO6	pin 90	DO7
pin 41	DI2	pin 91	DI4
pin 42	DI3	pin 92	DI5
pin 43	DI7	pin 93	DI6
pin 44	SMI	pin 94	DI1
pin 45	SOUT	pin 95	DI0
pin 46	SINP	pin 96	SINTA
pin 47	SMEMR	pin 97	SWO
pin 48	SHLTA	pin 98	SSTACK
pin 49	CLOCK	pin 99	POC
pin 50	GND	pin 100	GND

The physical realisation of the bus concept is the 'motherboard'; this is a printed circuit board with 100 copper traces running parallel down the board, usually

50 per side. At intervals holes are drilled through the board, perpendicular to the direction of the copper tracks, and a card connector fitted and soldered to the board. Typical motherboards have capacities for 10-20 S100 cards. The edge connectors for the cards have 50 contacts per side (double sided) on 0.125" centres. The card size (10" x 5.3") is now an industry standard.

The motherboard must be carefully designed if the system it is used in is to function reliably. In particular, it must have good shielding from external radiation to avoid corruption of data, also it should interleave the signal tracks with ground tracks to avoid crosstalk between adjacent lines.



An active terminator is also desirable: when the CPU or other device sends data onto the bus, transmission line effects will take place unless the bus is properly terminated; this can be avoided by the use of an active termination network, which additionally is used to help pull the lines up or down with the signals, thus speeding up transition times and increasing the noise immunity which is vital in a system using many cards loading the bus. An excellent example of motherboard design, which has all these features, is the Wunderbuss by George Morrow (see photograph).

Peripherals

There are large numbers of peripheral devices available to plug into the S100 bus and the following paragraphs deal with the main classes of cards available.

As mentioned previously, the S100 bus was designed around the Intel 8080 microprocessor; however, there are now numerous manufacturers offering alternative CPU's. The best-known alternative is of course the Zilog/Mostek Z-80 chip with which the 8080's instruction set is upward compatible. The Z-80 offers 158 instructions compared with the 8080's 78 and is obtainable in a 4MHz version (Z-80A) which is twice as fast as the standard 8080A. Other MPU's implemented for the S100 include the 6502, 6800 and TMS 9900. Although the S100 is nominally an 8-bit bus, one manufacturer, Alpha Microsystems, has gone to the trouble of interfacing a very powerful 16-bit microprocessor to the bus by multiplexing the 16-bit data down the 8-bit lines. This is totally invisible to the user, and gives phenomenal throughput due to the powerful architecture of the 16-bit CPU.

Random-access (read/write) memory is an essential part of every computer and the S100 bus system supports a variety of types. There are more suppliers of S100 bus compatible memory cards than any other type of card, and so the end user has a very wide choice. Board capacities vary from 4Kbytes (4x1024 8-bit bytes) to 64Kbytes. Boards also vary in their speeds (access time) and power consumption (heat dissipation). A typical board in use today is the Econoram III 8Kbyte dynamic memory; this has an access time of under 250ns and dissipates only 6W of power.

Read-only memory (ROM) boards are also popular; these are non-volatile, i.e. they do not lose their data when the computer is switched off, and hence are used to store bootstrap and power-up routines. The larger sizes are also used to hold systems software programs, e.g. monitors, editors and interpreters. They are useful in computers with slow program storage media, e.g. cassettes or paper tape, since then your 8K Basic does not have to be read in every time the system crashes or is switched off; this is the approach taken in the Commodore PET. In a system with fast mass storage like magnetic disks, it only takes a few seconds to load large programs anyway so large quantities of ROM are unnecessary. A small amount to hold the system bootstrap is nevertheless almost essential. The boards on sale are normally known as EPROM boards; EPROM is an acronym for Erasable Programmable Read-Only Memory; data can be programmed into specified locations by high voltage electrical pulses and later erased by ultraviolet light; hence it is desirable that an EPROM board should have an on-board programming circuit (a "PRAMMER") to ease the programming process.

Floating Point Operations

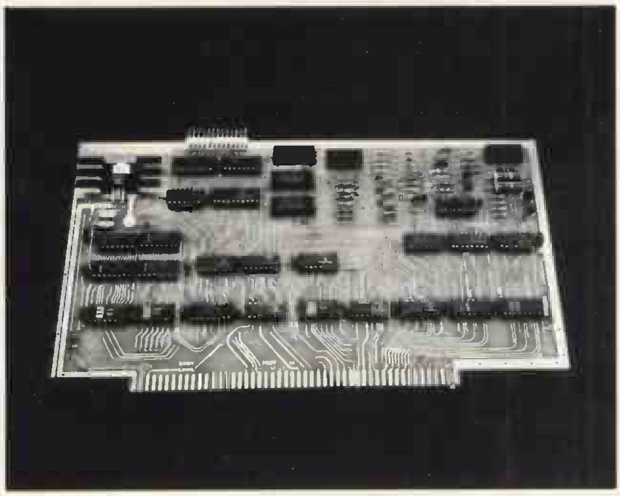
For certain applications involving a large amount of calculation, for example in science and engineering, the time taken by the computer to perform floating point operations can become significant. These operations (+, –, *, /) have to be implemented in software and this can be very time consuming for long computations. This difficulty is overcome with the S100 compatible floating point board by North Star. This performs the basic operations up to 50 times faster than the best software implementations and can lead to an amazing improvement in system throughput for scientific and engineering problem solving.

Input/Output

Now we will turn our attention to input/output in the microcomputer based on the S100 bus. The first requirement is usually some means of communicating with a terminal, either a visual display unit (VDU) or a teletype (TTY). These devices send and accept data as a string of bits in series and hence need what is known as a serial I/O port. Two different electrical interfaces are in common use, the V24/RS232C and the 20mA current loop; a good board should be able to produce both types of signal, however, the trend is away from the latter towards the former for various technical reasons.

The speed at which the interface communicates with the terminal is measured in bauds; in this context, one baud is one bit transmitted per second. Typical speeds are 110, 300, 1200 and 9600 baud. These represent 10, 30, 120 and 960 characters transmitted per second, and most S100 serial I/O boards can run at these speeds. The board ideally should have two serial ports so that two serial devices, e.g. a VDU and a printer, can be in use simultaneously. A useful addition found on most boards is a parallel I/O port. This is used for parallel devices such as paper tape readers, stand-alone printers and keyboards.

Mass storage of data, text, programs, object code and the like is necessary in all but the smallest microcomputer systems. The cheapest and most popular method at the moment is to use an ordinary £30 cassette recorder and an S100 audio cassette interface board. This combination stores and plays back data at speeds from 30 - 120 bytes (characters) per second. The most common recording standard is the Kansas City ("Byte") standard at 30 ch/sec (300 baud), which is very reliable.



The Morrow Speakeasy

A typical board using this format is the Morrow Speakeasy cassette I/O board: this will drive three cassette recorders with individual motion control. This enables the board to switch the motor of the recorder on and off, via the 'remote' socket on the recorder, after reading or writing a file on tape so saving tape space; this is quite important as only best quality tape should be used to avoid data errors. Total storage capacity will be about 50Kbytes per C60 side.

Although cassette storage is cheap, it is somewhat slow. At 300 baud, reading in an 8K Basic, for example, would take over four minutes. If data of this quantity had to be read in repeatedly, say ten times a day, then three-quarters of an hour would be wasted. The solution to this problem is in the use of magnetic disks. Most types of disk system can read or write 8K in under two seconds, over one hundred times faster than cassettes.

Disk storage

Magnetic disks come in three sizes or types: the smallest known as minifloppies, the intermediate known as full size floppies (or diskettes) and the largest known as hard disks; capacities are: minifloppies 80 - 400 Kbytes, full size floppies 250 Kbytes to 2 Mbytes and hard disks 10 Mbytes upwards.

A disk system consists of a controller board which fits into the S100 bus motherboard, with a cable leading to the disk drive which contains the motor, read/write head and associated electronics. As can be seen the S100 bus user has a wide choice of disk storage; the options available on each type of drive (like double density recording) widen the choice further to enable the user to choose a configuration precisely tailored to his needs. Unfortunately, pressure of space makes it impossible to describe all these possibilities; no doubt a future article in this journal will explore the subject in detail.

Now we come to some of the more unusual S100 compatible boards. For certain applications, the conventional VDU and its standard ASCII character set is inadequate. The solution here is to use an S100 compatible video monitor card with a video monitor. Now, the display of computer graphics is possible: applications include foreign language character sets, graph plotting, 3-D displays and games like Star Trek.

Process Control

Many of the microcomputers in use in industry are used for process control. Here the computer senses the state of the external world and causes action to be taken under control of its resident program. A card to do this via the S100 bus is the Mullen Optoisolator and Relay control board. This has eight inputs, each via an optoisolator, and eight outputs via relays. The status of the inputs may be sampled by a simple IN port instruction and the relays may be set by an OUT port instruction, one bit to one input or output.

The port address is selectable by a DIP switch on the board, hence N (N<257) boards can be slotted into the computer to control Nx8 devices in the outside world. In a similar manner, other boards to perform digital to analogue and analogue to digital conversion are available which can sense and generate voltage levels under computer control.

Three highly specialised examples of the latter types of card are speech recognisers, speech synthesisers and music synthesisers.

The Heuristics Model 50 speech recogniser is capable of recognising between 30 and 60 words spoken into a microphone connected to the card. The card digitises the speech input at specified intervals and this information is stored in the memory of the computer; typically 60 bytes of data are produced for each word. Then a software pattern match program compares the input speech pattern to the patterns already in memory; the closest match gives the answer. The effectiveness of this board is obviously dependant on a good algorithm for the pattern matching operation.

The Ai Cybernetics Model 1000 speech synthesiser is an S100 compatible card which generates synthetic speech. It is fed a stream of ASCII characters representing speech phonemes via an OUT port instruction and is very simple to program. The board circuitry is an analogue model of the human vocal tract driven by various oscillators and noise generators selected by an on-board ROM chip. The intelligibility of the system is low at first, but one soon learns to follow its unique accent.

The Newtech Model 6 music synthesiser board, with on-board loudspeaker, plays monophonic music; it sounds rather like an electronic organ and is driven in a similar manner to the Ai board. It is an interesting device for use in the home computer environment: have the computer play "For He's a Jolly Good Fellow" when you win at noughts and crosses, or the theme from Star Trek? . . .

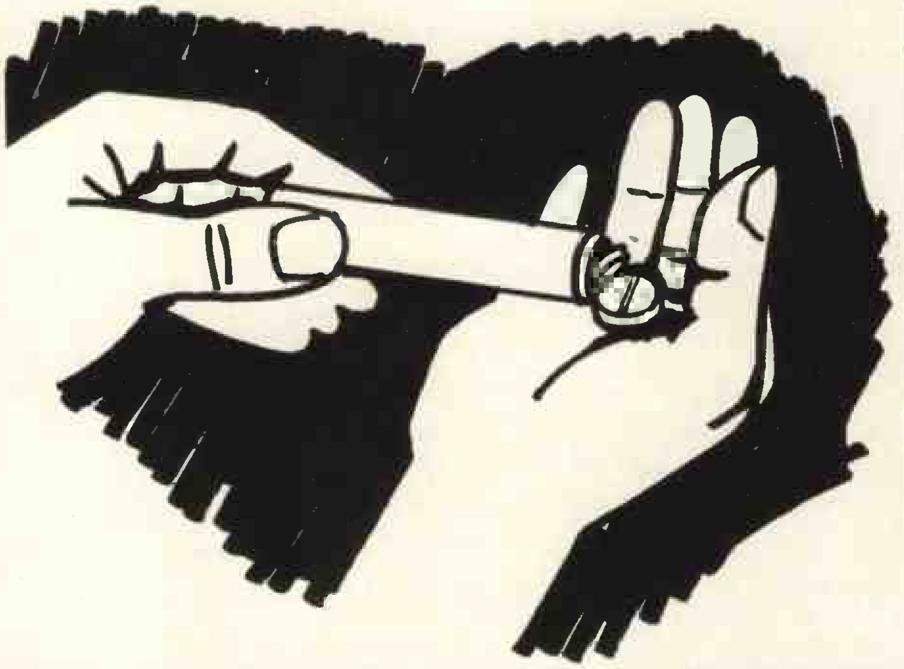
The last board to consider here is the S100 logic analyser. Microcomputer systems, like everything else, go wrong sometimes; the task of finding the problem in an S100 bus system is reduced by custom designed logic state analysers which plug into the motherboard like any other card. Signals on all the bus lines can be monitored and displayed on an oscilloscope as an 8-bit wide binary pattern, sixteen levels deep in time; this enables addresses, data, instructions, I/O ports and the like to be observed in real time, in the data domain. Selected binary patterns can be used to trigger an oscilloscope in the conventional mode so that hard to catch stitches can be observed and remedial action taken.

In conclusion, I hope this article has demonstrated the merits of the S100 bus; although no bus structure can be ideal for every application, the S100 has shown itself to be, by its adoption by over 150 manufacturers, a popular base for developing powerful and reliable computing systems.

References:
1. Intel 8080 Microcomputer Systems User's Manual.
2. Proceedings of the West Coast Computer Faire, 1978, pp 345-361.
3. Interface Age, June 1977, pp66 - 78.

Microcomputers and Mental Health

John A. Rope



When people who had psychiatric problems were locked up and forgotten, their 'care' was a simple if somewhat grim matter involving little skill and not much identifiable knowledge. Today, fortunately, the situation is very different. There is an immense array of new knowledge, techniques for treatment and rehabilitation of psychiatric patients. The use of this knowledge is in the hands of highly skilled personnel – doctors, nurses and, like myself, psychologists. Improvements in mental health care therefore would seem to involve large increases in these skilled personnel, so that more people may have proper attention. There, however, is the rub. The National Health Service employs a large number of people, there are branches in every town and wages alone are an enormous bill. In the present economic situation, little if any improvement in Health Service care is envisaged and that must mean that increases in staff of any kind will be minimal. Mental health has long been a Cinderella of the health field and Cinderellas rarely do well in an economic crisis.

I wish to propose a partial solution to the problem. I believe that much can be done in the way in which our present mental health resources are managed, and the technological backup they are given. By management I mean how the individual therapists or teams utilize the available time and information. The technological backup to which I refer consists of aids to problem formulation and solution, information control and retrieval and personal resource planning. It will be obvious to anyone who knows their capability that I have just outlined a role for the microcomputer. Few, if any, technological innovations could have arrived at such an opportune time but it remains a major challenge to the mental health professions to take advantage of this early period of microcomputer development so that we may influence its development in terms of software and suitable peripheral I/O devices.

This proposal is by no means a straightforward one. There are many prejudices in both psychiatry and psychology against not only devices such as computers but to new or different ways of thinking. Recently, at the Day unit of our hospital, we replaced the previous open referral system with one involving thorough pre-assessment, problem oriented records* and an adequate therapy and review system. When we described the "systems approach" to the psychiatric tutorial, the comment of an otherwise open minded consultant was that problem solution in psychiatry was an art and that "the next thing you know, they would be suggesting the use of computers to do medical diagnosis". He was less than pleased when I informed him that not only did computer diagnosis exist but was consistently more reliable and accurate than human diagnosis. I did not tell him that I would have introduced a computer into the system if I could have found someone to provide the money!

This sort of attitude contrasts sharply with that in general and surgical medicine where X-ray equipment is ancient history, electron microscopes are old hat and EMI scanners with their attendant capital and recurrent costs are the latest thing.

The prejudice is always well reasoned of course. The most general argument goes like this: psychiatric problems are largely interpersonal and social problems and need to be solved with a lot of face to face discussion.

*Problem oriented records cross reference all the daily records on a patient with an agreed problem list stating concisely what is wrong with him or her. These are a major improvement on traditional methods of record keeping, which can be difficult to understand.

Some versions of psychotherapy, for instance 'behaviour therapy', emphasise the need for the individual to learn from experience outside the therapeutic relationship, even so there is a fundamental need for human contact and understanding. It is not surprising therefore that many psychotherapists see the introduction of a 'machine' into the process as a hindrance, even a retrograde step.

Let us try and dispose of this problem first of all, without it we may consider the issue more soberly. To do so, we could do no better than to borrow a notion from that eloquent mathematician, the late A M Turing[1]. When considering the problem of whether a machine could be capable of thought, he suggested that a good test would be to question a man and a machine (a computer) simultaneously and remotely via a neutral communications device such as a teletype. If then the person posing the questions cannot tell which is which from the answers, we may accept that the machine can think (at least as well as the man). Suppose now that the computer is used as an extension of man's brain in psychotherapy, so the scheme shown in Figure 1A below is replaced by that shown in Figure 1B.

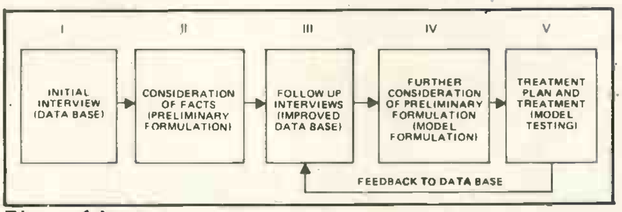


Figure 1A

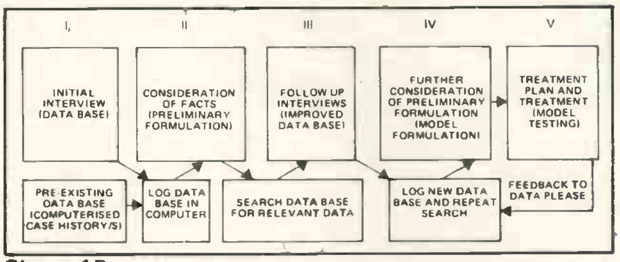


Figure 1B

What would the observer, in this case, the individual being treated, notice were he or she able to compare the two procedures? Not much, I suspect. They may notice that the therapist at stages III and IV seems to have a more thorough grasp of the facts of the case and to ask follow-up questions of a different or more penetrating nature, but as far as the observer is concerned, the primary relationship is maintained.

What if we now reveal to the person concerned that a computer is involved and, throwing caution to the winds, invite him or her to interact directly with it? Suitably programmed, the computer might well conduct much of the substance of the initial interview. This is not irresponsible day dreaming! As long ago as 1967, J Colby and H Enea[2], two American researchers into machine intelligence, programmed a computer to imitate a psychotherapist giving an interview. All the evidence indicates that, seated at a terminal keyboard and interacting with a computer which is behaving in this way, patients respond to it as if it were a psychotherapist. Furthermore, they seem to enjoy the experience. Reality seems to have moved very close to Turing's ultimate test of machines 'thinking' and it seems reasonable to suppose that computers, particularly microcomputers, may have a role to play in psychotherapy and may extend our horizons as well as standing in our place in certain time-consuming tasks.

The following are the areas where I foresee a future for microcomputers used on a personal basis by therapists and therapy teams. The list is by no means exhaustive and others with different perspectives and problems may be able to extend the list manyfold. I have tried to limit myself to those areas of most general use, irrespective of therapeutic orientation or professional training.

1) Data gathering, collation and retrieval (the psychiatric interview).
2) Model formulation and manipulation for the purposes of description and prediction (case diagnosis/formulation and treatment).
3) Treatment itself.
4) As a personal management tool.

These four areas are dealt with in more detail below:

1) *Data gathering, collation and retrieval*

As I have already indicated, microcomputers could be very directly involved in the gathering of data from the patient but the data base also consists of all that the therapist has learned from similar cases. Nowadays, so much is being written of interest and relevance to the medical sciences that indexes of scientific literature such as the 'Index Medicus' have had to be computerised to keep pace with the volume of material. Even at a personal level, it is possible to amass large quantities of information about various aspects of a particular disorder. A personal microcomputer would not only provide a suitable means of indexing and storing such information but also in collating relevant aspects, a task performed with varying efficiency by humans.

2) *Model formulation and manipulation*

All science is concerned with models in the sense that any theory, proposition or representation of reality may be considered to be a model, even if it takes conceptual rather than physical form. This area is the most speculative of the possibilities I am going to deal with. As far as I know, the methods for computers to use have yet to be developed. What follows is therefore based upon preliminary thinking of my own and it is not claimed to be authoritative.

Models in the psychiatric field are diverse and often seen by their practitioners as incompatible. For the most part, they are all conceptual i.e. held in people's heads or via language in books and non-systematic* in that even where they claim to cover the whole individual, they do

*I use 'systematic' here in the systems theoretic sense. That is, the human being should be described in terms of all the major influences on his or her behaviour, internal and external, in a way which specifies their interaction over time. I avoid the use of the more usual word 'dynamic' because in psychiatric parlance this usually refers to psychoanalytic oriented ideas.

not specify exactly how important components of the model interact over time. This frequently leads to non-linear-interactive relationships being simplified to fit linear-atomistic models which are easier to handle. So that two milligrams of antidepressant drug are often seen as twice as good as one milligram and psychiatric problems are isolated from the other aspects of the patient's existence which are not obviously connected to the occurrence of the problem. Evidence from models of other behaving systems such as those produced by Jay Forrester[3] indicates that startling effects may be produced on the system output by the cumulative effects of apparently minor parts of the total system.

Microcomputers seem to be the best available aid to both the research that is needed to develop the more complex models necessary and to exploit their potential. More immediately, there is the possibility of integrating the apparently incompatible models in use at the present. I spend much of my time doing this, and a microcomputer would allow a more sophisticated integration given a suitable set of concepts and programming language.

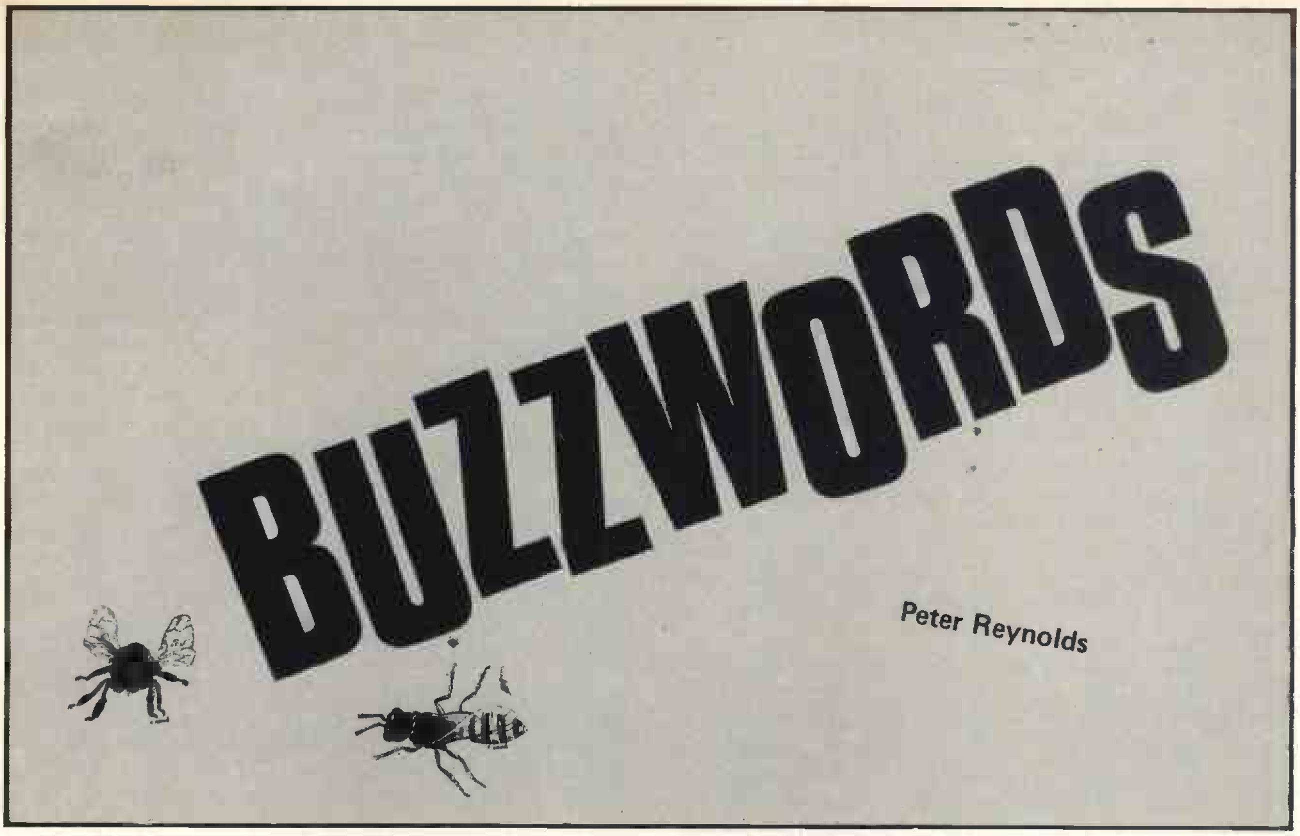
The eventual framework for the formulation, manipulation and evaluation of psychiatric models must lie in the methodology of General Systems Theory. Since the late L Von Bertalanffy[4] first indicated the need for a 'science of models' to express the complexities of interacting systems, more has been achieved in the areas of business and economics than pure and applied science. Forrester and his colleagues at the Massachusetts Institute of Technology have developed modelling techniques of extreme sophistication to describe economic systems ranging from those of companies to the economy of the world. Models for psychiatry will have to be different in form and concept from the economic models mentioned. They will have to deal with probablistic outputs rather than flow outputs from instance and interactions between system components may be in terms of excitation and inhibition rather than numbers of components or orders as is the case in economic systems. They are therefore likely to be more, not less, complex models and will still require the power of a computer to make them understandable.

3) *Microcomputers and treatment*
At present the most likely area of microcomputer involvement is that of gaming simulation. The games used are not necessarily different, in many respects, to the games sold commercially for pleasure, except that they are specifically designed to simulate an aspect of reality, of which we wish to give people some experience for the purpose of teaching. Many readers will be familiar with the idea of business games, aimed at teaching managers to handle simulated companies. Similarly, if our problem is to help someone who has difficulties in dealing with others, then we may consider developing a game to teach the rudiments of social behaviour. The more complex the game, and the more detailed its assumptions, the more it will be necessary to commit these to the memory of a microcomputer which can then monitor the progress of anyone playing the game.

4) *The microcomputer as a personal management tool*
Personal management applications of microcomputers can range from anything to do with the storing of case records, with the obvious implications for privacy and confidentiality, to patient scheduling and forward planning using software packages providing such things as 'Monte Carlo' models or 'critical path' analysis, tailored to the needs of the individual or small team. Small scale management innovations which might be possible using such techniques can be very effective. The reorganisation of the Day Unit which I mentioned above has resulted in a 25% reduction in patient numbers and a 30% reduction in patient attendances. The resulting improvement in patient care astonishes even us. The year's saving in ambulance use alone would pay the salaries of all involved for more than a year. However, we have reached the limit of what can be achieved by sheer reorganisation. Further improvement would require more complex and far-seeing management skills than we possess. It is equally important that the control of this remain in the hands of those clinically responsible and a microcomputer with suitable software backup could fill this need.

These then are the areas where I foresee personal microcomputers playing a role in the field of mental 'ill' health in as far as that term is defined. It should be added that there is also the huge field of promoting and maintaining good or positive mental health. Imagine your own microcomputer programmed with your own personal parameters and advising you on your best course of action for the near future to maintain or achieve a state of calm or happiness. Eventually it could do us out of a job! Not for some time to come perhaps, but it is a happy irony that in view of the dire predictions of microcomputers putting us all out of work, they may also allow us to live truly creative, happy and above all healthy lives.

1	TURING, A M	Can Machines Think? in Fiegenbaum & Feldman (Eds): Computers and Thought. New York McGraw Hill, 1963
2	COLBY, J and ENEA, H	Heuristic Methods for Computer Understanding of Natural Language in the Context Restricted On Line Dialogue Mathematical Biosciences 20, 1967
3	FORRESTER, JAY W	Understanding the Counterintuitive Behaviour of Social Systems in Beishon, J & Peters, G: Systems Behaviour pp 200-217. London, Harper & Row, 1972
4	VON BERTA-LANFFY, L, HEMPEL, C G, BASS, R E & JONAS, H	General Systems Theory: A New Approach to the Unity of Science, 1—VI Human Biology, 23, 302-61, 1951



A to D Converter – Analogue to Digital converter. A device, typically an integrated circuit, which gives a digitally coded output whose value is proportional to an analogue input such as a varying voltage. For instance a digital thermometer comprises, in essence, an analogue sensor, an A to D converter and a digital display.

Absolute. Pertaining to a fixed address as opposed to a relative address; see *absolute address.*

Absolute Address. The description of a location in a computer's working store which identifies it explicitly and unchangingly in machine language, without reference to any other location. An absolute address can be expressed as a number, for example, 5000, signifying the five-thousandth storage location. See *address* for an illustration of the distinction from *modified* and *relative* address. Synonymous with *specific* address. See also *direct* address.

Absolute Coding. Program coding which uses machine instructions in which *absolute addresses* are employed (as opposed to relative addresses). Such a program avoids the use of a modifier register or B-register.

Absolute Error. The magnitude of an error, disregarding its direction or sign. An absolute error of 10% in a measured kilogram could mean 1100 grammes or 990 grammes.

A.C. Alternating Current (abbreviation).

ACIA (for Definition see PCW Issue 3)

Acceptance Testing. The tests carried out on a computer by, or on behalf of, a customer to demonstrate that the equipment is in proper working order before it is handed over. Acceptance tests are commonly based on special programs which are designed simply to put the hardware through its paces and possibly to run continuously for sufficient time to indicate the reliability of the equipment and the incidence of error. A more useful trend also associates various commercial programs and software with acceptance tests and a computer may be required to pass tests specifically employing the routines that the customer will use in daily practice.

Access. To establish contact with an area of computer store for the purpose of reading or recording data there.

Access Time. The period elapsing before the desired record can be selected in some sequential or random access storage device. In the case of a sequential record, such as magnetic tape, access time will comprise the time required to start and stop the tape transport, together with the running time required for the appropriate length to pass the read/write heads. This "access" time could be as long as the time required to run the tape from end to end if, by any chance, the data searched for happens not to be in the record.

A magnetic disc (floppy or rigid) will have an access time equal to the latency time, that is, the time taken for the desired information to come round to the point where the reading head is situated. On average this will be the duration of half a revolution of the disc. Where moving heads are employed to cover several tracks, the time of the head movement to the appropriate track must be added to the latency.

***Accumulator** (for Definition see PCW issue 3)

Accuracy. Degree of freedom from error, that is, the extent of conformity to truth or to some rule. Accuracy may be contrasted with precision – for example, if 22/7 is calculated as .314285713 . . . it could be expressed with complete accuracy to three places of decimals as .314 and with more precision to six places of decimals, but the expression .314285 would be inaccurate (the corrected value, by normal conventions, being .314286).

Adaptive System. A computer control system with parameters which are adjusted automatically to produce the best response to each demand. For example, a program may be arranged to read in a certain quantity of data on each cycle. An adaptive control system might recognise the condition when a block of data is blank and go on to read in the next block immediately.

Adder. Circuitry to perform addition; the result may be transferred to an *accumulator.* See *half-adder and full-adder.*

***Address** (PCW Issue 3)

Address bus. The common electrical path, designed to carry address data, in a computer. In part it is likely to comprise multi-way ribbon cable connecting one sub-chassis with another.

***ALGOL** (for Definition see PCW Issue 3)

Algorithm. A set of explicit rules for the solution of a mathematical problem. For example, a fully defined statement of procedure for calculating the sine of an angle to a stated degree of precision. It derives from the name of a mathematician of the 9th century A.D. (Al Khwarizmi).

Alpha. Contraction of alphabetical; generally signifies letters of the normal alphabet and punctuation as opposed to numeric characters.

Alphabet. An ordered set of unique characters. In the EDP (Electronic Data Processing) sense "alphabet" can include figures; for example, the binary alphabet comprises 0 and 1.

Alphameric. Contraction of *alphanumeric.*

Alphanumeric. Characters including both letters and numerals, and usually other symbols as well, such as mathematical signs and normal punctuation marks. A popular subset of ASCII comprises 64 characters, half the full set (see illustration).

***ALU** (for Definition see PCW Issue 3)

Analogue A system of representing quantities by analogy with physical variables, for example, electrical voltage. The usual motor car speedometer is an analogue instrument representing speed by proportionate movement of an indicator. The milometer is a form of digital recorder. Computers based on the analogue principle are normally restricted to such functions as process control. The *digital* computer has a virtual monopoly of commercial applications.

AND Gate. A basic piece of computer circuitry which effects the AND or logical multiply operation. The element has two or more input connections and a single output wire which gives a signal if, and only if, all the input wires receive a simultaneous signal.

AND Operator. Boolean operator; synonymous with *logical multiply*. Imagine two statements P and Q. P AND Q will be true if, and only if, both statements are true. The AND operator may be represented by a centred dot (P·Q); by the multiplication symbol X (P X Q); by the logical product symbol (P $\wedge$ Q); or by no sign at all, simply (PQ).

Architecture (for Definition see PCW Issue 3)

Arithmetic Check. Check of a computation by making use of its arithmetic properties. For example, multiplication of A X B can be checked by comparing it with the result of B X A.

Arithmetic Operation. A computer operation in which ordinary basic arithmetic operations are performed on data; for example, addition, subtraction, multiplication and division. Arithmetic operation may be contrasted with *logical operation.*

Arithmetic Shift. Multiplication or division of a number by a power of the base of notation. In the case of the usual decimal numbers (where base of notation is 10) each shift to the left multiplies the number by 10—thus a shift of three places left multiplies by 1000. In the case of binary numbers a shift of one place to the left multiplies the number by two; a shift of three places to the left would multiply by eight. Similarly, each successive shift of a binary number to the right would divide the number by two, four, eight, sixteen, and so on, just as such a shift of a number in decimal mode would divide it by 10, 100, 1 000, and so on.

Arithmetic and Logical Unit (ALU). That part of a computer's central processor which holds the circuits that perform arithmetic and logical operations.

Array. An arrangement of data is a row or matrix, so that any element of data in the array may be identified by as many subscripts as the array has dimensions. A street of houses is an example of a one-dimensional array where street (5) indicates the fifth house in the street. A multi-storey hotel is an example of a two-dimensional array where Hotel (12, 18) would identify the 18th room on the twelfth floor.

- * ASCII
- * Assembly Program (For Definition see PCW 3)
- * Assembly Language

ASR. Automatic Send and Receive. Used mainly of sequential printers (such as the Teletype) with built-in paper tape or magnetic cassette whereby the terminal can receive and record messages when unattended and can also transmit, more or less unattended, using a tape which has been prepared in advance.

Associative Processing. Computing where data or program instructions are identified not by conventional addresses but by the actual content of that data or those instructions. For example an editing instruction might require Ms to replace Miss or Mrs wherever they appeared in a file of names.

Asynchronous Working. A mode of computer operating whereby each operation is started by a signal given on completion of the previous operation, or on the equipment required for the next operation becoming available. This contrasts with *synchronous* working, in which the sequence of operations is controlled by a master clock without regard to the completion of earlier operations. It is possible for a synchronous sub-routine to form part of a larger operation which is working asynchronously.

Audio Response. Form of computer output, like the GPO talking clock TIM, in which meaningful information is assembled from a 'dictionary' of pre-recorded words and phrases.

Authority List. A computer-held record of the code identification of individual terminals in a multi-access system. The list might stipulate, for instance, that some terminals could be used to alter records but not others and the computer would apply these inhibitions automatically. Authority lists can also be used to discriminate between individual operators of remote terminals where these people identify themselves by *passwords* or even by personal, machine-readable, identity cards which have to be presented to the terminal machine before operating starts.